10-1-2012

Electric vehicle charging and routing management via multi-infrastructure data fusion

Christopher Decker

Electric Vehicle Charging and Routing Management via Multi-Infrastructure Data Fusion

By:

Christopher L. Decker

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science
in Computer Engineering

Supervised by
Associate Professor Dr. Shanchieh Jay Yang
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
October 2012

Approved By:

Date:

Dr. Shanchieh Jay Yang
Primary Advisor – R.I.T. Dept. of Computer Engineering

Date:

Dr. Andres Kwasinski
Primary Advisor – R.I.T. Dept. of Computer Engineering

Date:

Dr. Clark Hochgraf
Secondary Advisor – R.I.T. Dept. of Electrical, Computer and Telecommunications Engineering Technology

ABSTRACT

Electric Vehicle Charging and Routing Management via Multi-Infrastructure Data Fusion

Christopher L. Decker

Supervising Professor: Dr. Shanchieh Jay Yang

The introduction of Electric Vehicles (EVs) has placed a strain on the aged and already overworked electrical grid. With each EV requiring the same amount of power as 3 to 140 single family homes, depending on how fast the charge occurs, measures need to be taken in order to protect the electrical grid from serious damage. The electric grid renovations proposed by the U.S. department of energy, commonly referred to as the smart grid, could help accommodate an even greater EV penetration. The introduction of the smart grid and other cutting-edge technologies create the potential for applications which provide new consumer conveniences and aid in the preservation of the electrical grid.

This thesis aims to create one such application through the production of a prototype system which takes advantage of current and in-development technologies in order to route an electric vehicle to the closest and least detrimental charge station based on current conditions. Traffic conditions are assessed based on data collected from both ITSs (Intelligent Transportation Systems) and VANETs (Vehicle Ad-hoc Networks), while grid information is gathered through the early stages of the Smart Grid. The system is hosted in a cloud environment base on the current trend of offloading Information Technology systems to the "cloud"; this also allows for the advantages of a shared data space between sub-systems.

As part of the thesis the prototype system was put through a stress test in a simulated environment in order to both establish system requirements and determine scalability for use with larger maps. The system requirements were compared with the technical specifications of an off-the-shelf GPS routing device. It was determined that such a device could not handle routing with such extensive underlying data, and will require hosting the prototype in a cloud environment. The system was also used to perform a case study on charging station placement in the Greater Rochester area. It was determined that the current charging stations are insufficient for a significant number of electric vehicles and that adding even six stations would provide a greater EV operational area and provide a more uniform distribution of charging station usage.

ACKNOWLEDGEMENTS

I would like to sincerely thank my advisors, Dr. Shanchieh Jay Yang, and Dr. Andres Kwasinski for their guidance, support, and mentoring throughout the Thesis process.

I would also like to thank my committee member, Dr. Clark Hochgraf, for reviewing my work and providing invaluable advice.

Lastly, I would like to thank Kristin Atkinson for providing emotional support and proofreading this document, as well as Michelle Decker for her professional level expertise of the English language.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

GLOSSARY

A*- a computer algorithm that is widely used in pathfinding and graph traversal

AP (Access Point) - a device that allows wireless devices to connect to a wired network, such as the World Wide Web using Wi-Fi, Bluetooth or related standards

Cloud-the use of computing resources (hardware and/or software) that are delivered as a service over a network

EI (Energy Index) – The rate of battery depletion per unit of measure, typically time or distance.

EV (Electric Vehicle) – A vehicle, typically a car, which uses one or more electric motors or traction motors for propulsion

ITS (Intelligent Transportation System) – the application of advanced electronics and computer technology to automate highway and vehicle systems to enable more efficient and safer use of existing roadways

PHEV (Plug-in Hybrid Electric Vehicle) - a hybrid vehicle which utilizes rechargeable batteries, or another energy storage device, that can be restored to full charge by connecting a plug to an external electric power source

Smart Grid - an electrical grid that uses information and communications technology to gather and act on information in an automated fashion in order to improve the efficiency, reliability, economics, and sustainability of the production and distribution of electricity.

SOC (State of Charge) –The percentage of power left in the battery. Equivalent to a fuel gage

SUMO (Simulation of Urban MObility) – A commonly used traffic simulator

VANET (Vehicle Ad-Hoc Network) - a technology that uses moving cars as nodes in a network to create a mobile network

VANET penetration- the fraction of vehicles with VANET capabilities

VM (Virtual Machine) - completely isolated guest operating system installation within a normal host operating system

1 INTRODUCTION

The increase of gas prices and the concerns over climate change are helping to increase the popularity of "Plug in Hybrid Electric Vehicles" (PHEVs) and pure "Electric Vehicles" (EVs). The primary problem is that charging these vehicles overnight roughly requires the same amount of power as a small house. Currently in the United States, transformers are designed to provide power to 3-5 houses [1]; if the average family buys two EVs, these transformers would suddenly have to support the equivalent of 9-15 houses. Transformers are not designed to handle such large loads and would quickly fail. Replacing or remodeling the power grid in order to update these transformers would be time consuming, expensive, and most likely unnecessary as the transformers would still only support 3-5 houses during the day, the increase would only come at night. As it stands right now, power demand is predictable, during the day the demand is high in the business districts and centers of commerce, while in the evening the demand is high in the residential sectors [2]. It is unknown how this will change with the addition of EVs.

Eventually, consumers will not be content with charging their vehicles overnight and will demand the convenience and accessibility currently available with petroleum fueled vehicles, a quick re-fueling at conveniently located stations [2]. Currently a ten minute quick charge requires the equivalent of 140 small homes; such a large, dynamic demand is not possible with the current U.S. power grid. There are plans in development to create a "Smart Grid", which will allow for dynamic power routing; however the implementation of such a grid is still many years off. The most popular short term solution is that each station contains a battery; the station can then draw energy based on what is available from the grid, and rapidly discharge the battery when an EV charges. In order to obtain the same convince of modern day gas stations, either the EV operators would have to be informed which stations currently have enough power to charge their electric vehicle, or the station would have to be informed that a vehicle is on its way so it should start collecting power [3].

At first glance it is not obvious what the computing field can contribute to preventing damage to the electric grid, as it is very much a multi-disciplinary problem. Recently, renting data storage and computation capability from large data centers, known as "Cloud Computing", has become popular. This allows for easy data access and data sharing across infrastructures, such as Intelligent Transportation Systems (ITS), Vehicle Ad-Hoc Networks (VANETS), and the Smart Grid [4]. It is the responsibility of those in the computing profession to develop systems that use the data from these multiple sources in order to predict and respond to dynamic power flows as well as systems that advise customers where and when power is available. It is also the responsibility of Computer Scientists and Engineers to develop models and simulations to advise the Electrical, Civil, and Mechanical Engineers in their design of a new power grid model. One of the many tools available to Computer Scientists is the "Cloud".

Although there is no consensus of a formal definition of Cloud Computing, most experts agree that Cloud Computing refers to the applications delivered as services over the Internet, as well as and the hardware and systems software in the data centers that provide those services. Three models of cloud computing are currently popular: Infrastructure as a service (IaaS), where the company simply provides the hardware and all software is maintained by the user, Platform as a Service (PaaS), where the company provides Virtual Machines (VMs) to the users (e.g. Amazon Web Services), and Application as a Service (AaaS), where there company provides users access to a program such as Google Docs through their browser [5]. Regardless of the infrastructure most clouds are elastic; the amount of resources being allocated can be scaled dynamically. This is both useful in that a user does not have to pay the same amount during a dip in in processing as they do during a peak in processing, as well as in that it allows the Cloud provider to re-allocate unused resources to a different user.

The proposal to join all Clouds together in a federated model is rapidly gaining support in the research community [6]. This involves high speed data lines between the cloud data centers of different providers so that they may rent resources from each other. This has the added benefit of allowing programs running on different clouds to communicate with each other at incredibly fast rates, as if they were located on the same machine [5]. As more and more operations are outsourcing their IT infrastructures to the cloud, a new opportunity for cooperation and data sharing is forming. If corporations were to share their non-proprietary data with each other rather than keeping it to themselves, as is the current practice, solutions to numerous problems would become viable, including the EV charging situation. This thesis aims to provide a proof-of-concept prototype system to demonstrate that systems sharing data on the cloud would allow for applications not practical otherwise. The system includes sub-systems within the cloud which emulate data from three different sources: ITS, VANET, and the smart grid in order to prove this point.

The goal of this thesis was to develop a prototype system that advises the driver of a PHEV/EV of which charging station they should use based on traffic conditions, the amount of energy remaining in the vehicle, and the cost of electricity at the charging stations, representing the state of both the station and the electrical grid. The system takes into account both the convenience of the user and the needs of the grid. The objectives of the prototype system were threefold: to provide a proof-of-concept system to showcase that systems sharing data on the cloud would allow for applications not practical otherwise, to provide a system which can be used to analyze behaviors of EVs and the grid, and to provide a system which others can build off of in order to test their own work. Wherever possible, the system was designed in such a way that adding more data points would be trivial in order to facilitate future work.

A literature review was performed in order to select the best existing/in-development systems to collect data from. After these systems were selected, a deeper search was conducted in order to obtain a full understanding of these systems so that realistic data could be emulated. The SUMO traffic simulator [7] was selected to

emulate the data related to traffic patterns and the website OpenChargeMap [8] was selected to emulate the smart grid using real-world data. In order to emulate a real-world cloud environment, the sub-systems which supplied data-points where encapsulated in separate VMs within a small cloud.

The system was simulated under variations in traffic flow, number of cars, number and location of charging stations, and user preferences. Sets of simulations were conducted in order to determine system performance on the test platform, to demonstrate the systems usefulness in selecting charge station locations, and to prove the optimality metric behaves as intended. Lastly, future improvements and uses for the prototype system are outlined, and detailed instructions are provided for modifying the system.

The rest of this document is organized as follows: Chapter 3 gives an overview of a portion of the related/similar work found in the literature review, Chapter 4 provides a detailed system overview and provides justification for the selection of data points used, Chapter 5 provides a discussion of all collected results, and Chapter 6 concludes the document with an overview of possible future work and closing remarks.

2 BACKGROUND AND RELATED WORK

2.1 UNDERSTANDING THE PROBLEM

While charging, Electric Vehicles (EVs) pose a strain to the electric grid for which it is not currently prepared. In order to fix any problems in the grid created by EV charging, it must first be fully understood how electric vehicles, charging stations, and the grid interact. This can be done in two ways, observing current data before making extrapolations or making simulators in order to test various scenarios. While this research is being performed to fix the grid other researchers are exploring stop-gap measures to prevent grid damage until such a time as grid renovations can be completed and the smart grid brought online.

Until recently, fast charge stations were not readily available. Most research in the area was done using prototype systems such as those designed by Winkler *et al.* [9]. These prototypes were designed for teaching and research purposes rather than commercial use. Unfortunately, due to a lack of a fast charge standard or an agreed upon methodology, these prototype systems produced a wide range of varying results. With the release of the first public fast charge station in 2010, researchers now have a commercially available system to use for testing [10]. Alternatively, researchers can observe these systems in real world operation, in order to collect more viable data.

In addition to evaluating individual physical systems, many governments have performed case studies in order to evaluate how their electric grids would fare against a sudden influx of electric cars and to use as a guideline for EV based policies. The state of Vermont took it a step further and extrapolated the number of EVs which could be supported under various conditions [11]. They found that the Vermont electrical grid could support 50,000 EVs under normal operating conditions, 100,000 if the vehicles waited until midnight to charge, 200,000 if an optimal charging algorithm was used.

Simulations are the primary method of understanding the interactions between Electric Vehicles, Charge Stations, and the Electric Grid. The difficulty in creating an accurate simulation is the number of models which must be integrated; there must be an Electric Vehicle model, a Charge Station Model, the Electric Grid must be modeled, there must be a traffic model, etc. In an ideal system, models would be able to be swapped out in a plug and play fashion for comparison purposes, or to replace an inferior model.

Models for electric vehicles and charging stations will be discussed in later sections; two very different electric grid simulation models will be highlighted here. PMSS is a functional simulation model, it models individual components, their behaviors, and their interactions [2]. Its developers created the model as a black box program in order to make it easier to use; the user sets a variety of parameters for each node (think cluster of businesses, hoses, etc.) and an error signal indicates if any grid constraints have been violated. Yunus *et al.* take a completely different approach through the use of statistical modeling [12]. They use a Markov chain model to emulate system

level events and then use a bipartite network model to connect multiple systems. Although PMSS and other functional models are more accurate, statistical models provide faster results, a necessity in large simulations.

2.2 ELECTRIC VEHICLES

Electric Vehicles (EVs) are vehicles which use some form of an electric motor as a means of propulsion. Two main forms of EVs exist, pure electric and hybrid. Pure electric vehicles run purely on an electric engine, while the electric engine in a hybrid electric vehicle is supplemented by an internal combustion engine. A key advantage of electric or hybrid electric vehicles is regenerative braking and suspension; their ability to recover energy normally lost during braking as electricity to be restored to the on-board battery. This thesis focuses on electric automobiles that have the capability to hook their electric motors up to a base station for charging (Pure Electric and Plug in Hybrid).

Electric Vehicles offer numerous advantages and disadvantages, both to their users and to the environment. The primary advantage is cost, in spite of an initial higher price the cost difference between electricity and gasoline make EVs still economic to operate. For example, electricity at around US$0.10/kWh translates to an equivalent gasoline cost of about US$0.70 per gallon [13] (compared to $4.00 at the pump in September 2012). Additional advantages and disadvantages are included in Table 2.1.

TABLE 2.1

EV ADVANTAGES AND DISADVANTAGES

Advantages	Disadvantages
a reduction in petroleum dependency (side effect of a reduction of foreign oil dependencies)	long recharge times
reduction of CO2 emissions	a reduced range
use of the existing electric infrastructure for charging	a higher initial cost
a reduction of noise pollution	a slightly higher accident rate due to the lack of noise
and performance (acceleration, response, etc.) is likely to be on par if not better than similar conventional vehicles	and of course the strain on the electric grid produced from charging

Most technology used in Electric Vehicles is efficient and cost effective; however, EV battery and control technology still has a lot of room for improvement. The leading technology currently under investigation for use in EV batteries is Lithium Ion [13], it offers a balance between cost, durability, and performance. However, in their current state, Lithium Ion Batteries are incredibly costly to produce. Manufactures predict that a rapid development in technology and a discount from volume production will drive the price for these batteries down within the next couple of years [14].

In anticipation of the development of the smart grid, OnStar has started developing a series of APIs in order to allow an EV to interact with the grid [15]. OnStar already has a vehicle-to-vehicle/vehicle-to-base-station infrastructure in place, which it can leverage in vehicle-to-grid communication. They have developed user friendly smart grid capable applications, such as texts to remind car owners to plug in their cars, and delayed charging. In delayed charging the car charges during the selected interval, not the entire time it is plugged in. OnStar plans to implement destination prediction, allowing the grid to route power to the car's predicted destination, and Time-of-Use rates, allowing the vehicle to automatically determine the best time to charge itself. The API is freely available to grid-developers on OnStar's website.

For EVs, it is possible to model the distance a vehicle can travel as a function of the amount of energy available in the battery, maximum speed, and average speed it will be traveling [2]. This would be a relatively simple calculation, except other devices within the car (such as radios, windshield wipers, lights, etc.) also drain the battery, and external forces such as gravity and inertia act on the car. Maia *et al.* attempted to model an EV with a focus on energy consumption [16]. They modeled all forces acting on the car, including rolling resistance, aerodynamic drag, inertia, gravity, etc, and translate these forces into a cost of energy by incorporating velocity; finally they add a factor for AC drain in order before producing a measure of the amount of energy an EV is expending. They go further to model the amount of regenerative energy the car produces from breaking. They then use both of these results, the energy flowing into and out of the battery, in order to estimate the vehicles current state of charge (SOC), the current percentage of energy remaining in the battery.

Maia *et al.* proceeded to implement their model in the traffic simulator SUMO. They first validated their model against a circuit commonly used in consumer test reports and received results similar to those reported by actual EVs. They then implemented a circuit that ran a path in Coimbra City and found that the rate of the change of the EVs altitude was the most significant factor in determining the SOC depletion rate. This was due to the fact that a steep incline requires a significant amount of energy to traverse, while on a steep decline the EV recovers a significant amount of energy from regenerative breaking.

2.3 CHARGING INFORMATION

While most HEVs recharge through their gas engine or regenerative breaking, those that rely more on their electrical motor recharge by plugging in to the electrical grid. Most countries' power grids function by transmitting power at high voltages over long distances and then stepping the power down in numerous stages until it reaches household voltages (120V in the US and 220-240V in Europe). The U.S. power grid can be broken down into two sub grids: the transmission grid, and the distribution grid [17]. The transmission grid consists of the series of high voltage wires that transport power from power generators (power plans, wind/water turbines, solar panel farms, etc.) to far off cities and towns. The distribution grid consists of the low powered wires, substations,

hubs, and transformers that step down the power from the high voltage lines and disperse said power to the numerous homes and businesses within its responsibility. Various investigations have shown that an influx of electric cars would cause trouble in the distribution grid, long before they would affect the transmission grid. [11] [3]

2.3.1 SMART GRID

The power grid in most first world countries is a product of rapid expansion during the industrial area, and as such is poorly designed, old, and generally inadequate in the modern age. The proposed renovations, commonly referred to as the Smart Grid, would solve most if not all problems dealing with charging EVs. The Smart Grid is essentially an Internet connected electrical grid. It will allow for dynamic energy routing, predictive grid maintenance (rather than reactive), detection of outages (rather than relying on customers to call it in), the ability to inform appliances when there is a shortage and should shut down, the ability to inform EVs of the price of electricity to they can charge when it is cheapest, etc. [18]. Additionally, the implementation of this technology would likely require an overhaul of the distribution grid, which would solve most EV charging problems. The Smart Grid is a long way off; however, electric companies have yet to agree on a standard, it will also require a complete overhaul of the current electrical grid, for which no one is eager to pay.

Recently, the US Department of Energy released a vision statement/call to action which highlighted six characteristics of the smart grid which they feel need the most improvement [19].

1) Self-Healing - the smart grid must perform continuous self-assessments to detect, analyze, respond to, and as needed, restore grid components or network sections.
2) Motivate consumers to be active grid participants - the smart grid must inform it's users of current electricity prices and demand so they may make economic, environmentally friendly choices.
3) Attack Resistant - the smart grid must reduce physical and cyber vulnerabilities and recover rapidly from disruptions
4) Provide the level of power quality desired by 21st century users - new power quality standards will balance load sensitivity with delivered power quality at a reasonable price. The smart grid must supply varying grades of power quality at different pricing levels.
5) Accommodate all generation and storage options - the smart grid must accommodate all sources of power storage and generation in a plug and play manner, including green energy generators such as wind turbines and solar panels.
6) Enable markets to flourish - the smart grid must allow more market participation through an increase in transmission paths and aggregated demand response initiatives.

This thesis aids in improving point two by advising users as to the closest, cheapest charging station (also best for the grid), and point six as the system has the potential to advise in the placement of new charging stations in locations where they are needed and will be profitable.

2.3.2 FAST CHARGE STATIONS

Charging an Electric Vehicle at a station can take a number of hours due to limits in the amount of energy which can be drawn from the grid. In order to allow customers to quickly charge their EVs (in 15-20 minutes), Quick Charge Stations are being developed which are modeled off a modern day gas station. However, as stated earlier, quickly charging a significant number of EVs requires more power than the current electric grid can supply on demand, thus an energy storage system, typically a battery, is required. It is common practice to think of this battery as being similar to the large petroleum storage tanks located below most gas stations. This battery acts as a buffer between the station and the grid, absorbing peak and valley in demand which allows a quick charge station draws a near uniform charge from the grid. This allows the power companies to allocate a larger amount of power to the station because the power draw is easily predictable, typically enough power to charge two to three EVs without drawing on the station's battery.

This buffering is accomplished by accounting for three states of operation: high-load, middle-load, and low-load. While in its high-load state, the station cannot draw enough power from the grid to supply its current influx of customers; therefore it must supplement the power from the grid with that from its energy storage unit. In its low-load state, the station has few or no customers and can use the extra power from the grid to charge its energy storage unit. The third and least used state is middle-load, in this state the power draw to the station's customers is equivalent to the amount of power drawn from the grid [20]. It is common to eliminate the middle-load state in practice, as it can be modeled with either the high-load or low-load states.

Schroeder *et al.* performed an analysis on the potential for profit from fast-charge stations [21]. They found that at this stage, there is no possibility for profit from fast-charge stations. However, under the right future conditions charging stations could be more profitable than modern-day petroleum gas stations. Because of the lack of profit, currently fast-charge stations will only be purchased for one of two reasons; the first is as an added incentive to attract customers to businesses which have alternative revenue streams. The other possibility is that a patron who believes in the idea electric cars will make a purchase in order to support the venture. They believe that without a new incentive, fast-charge stations will quickly loose popularity. Nansai *et al.* presents the contrary opinion that the popularity of fast-charge stations has already reached a self-sustaining point and will continue to grow for as long as the popularity of EVs continues to grow [22].

2.4 INTELLIGENT TRANSPORTATION SYSTEMS

In addition to knowledge on the current state of the electrical grid, EV charge routing will also benefit from knowledge on current traffic conditions. Currently, numerous states and countries have sensor networks set up in order to monitor their roadways. These networks usually record three main data points: average speed along a road segment, the rate of vehicles entering and exiting a road segment (for example exit/entrance ramps for highways), and the average occupancy of a road segment [23]. The administering organization then feeds these data points into an Intelligent Traffic System (ITS); one such system is designed to monitor and predict traffic flow and administer the roadways in such a way as to obtain their maximum throughput. The ITS accomplishes this by controlling traffic lights, changing the direction of travel for variable lanes, and advising vehicles of alternative routes or available parking spaces. This is the functionality focused on in this thesis; however Table 2.2 contains a summary of the six major classifications of ITSs [24]. An algorithm that requires information on traffic flow would benefit greatly from the same data collected for ITS systems.

TABLE 2.2

ITS CLASSIFICATIONS

Category	Description
Traffic Management	Collects information from roadside detectors in order to manipulate traffic flow (discussed above).
Traveler Information	Advises users as to traffic conditions along road segments. Uses the same collection sources as Traffic Management. Also included GPS systems which use real-time traffic data.
Commercial Vehicle Operations	Used in large and medium companies to monitor the state of their delivery vehicles. This includes both technical information such as if the engine needs maintenance and tracking information such as speed and stop times
Public Transportation Systems	Uses technologies from Traffic Management and Traveler Information to improve mass transit, by informing users of route information, travel schedule and costs, real time information on transport systems. Some systems even give priority to mass transit through careful traffic light manipulation.
Vehicle Control Systems	Uses sensors to alert users of traffic conditions, collision avoidance, or even help take part in driving.
Rural Transportation Systems	Variations of the above categories specifically designed for the sparse density found in rural areas.

2.5 VEHICLE ADHOC NETWORKS

A Vehicular Ad-Hoc Network (VANET) is a system which uses moving cars as nodes in order to create a mobile network; it is predicted that eventually VANETs will be the primary data collection method for ITS systems. VANETs turn every participating vehicle into a wireless router or node, allowing vehicles approximately 100 to 300 meters from each other to connect and, in turn, create a network with a wide range. These networks are always changing, getting larger or smaller; new networks are forming while others are breaking up. Due to the fact that there might not be a single network spanning a packet's source and destination location, vehicles also have a store and forward feature; as a vehicle leaves one network, it acts as a carrier and transports the packet to a different network. Additionally, stationary roadside access points have been proposed to give these networks access to the internet proper, allowing for communication with traditional application.

As shown in Figure 2.1 the popularity of research in the field of VANETs has risen dramatically since the first paper was published in 2005. Because the field is relatively new, there are numerous directions of research; such as protocols, transmission technology, vehicle/network integration, and applications. Two of the most popular include simulator development, such as the one presented by Martinez *et al.*, and packet forwarding protocols, such as those presented by Chuah *et al.* [25][26]. VANET simulators are being created by fusing two already existing simulators: traffic simulators, such as SUMO and VISIM, and packet-level network simulators, such as NS-2 and QUALCOM [25]. This poses numerous challenges, such as establishing feedback loops, which allow the network simulator to receive location/speed information from the traffic simulator and the traffic simulator to react to information received from the network simulator. A completely different sub-area in VANETs is routing protocols. When re-transmitting or carrying packets, it is ideal that progress is being made towards the packet's destination. This can be difficult with such a dynamic network where routing tables are difficult if not impossible to produce. Jeong *et al.* developed a probabilistic model based on a car's current direction and speed [27].

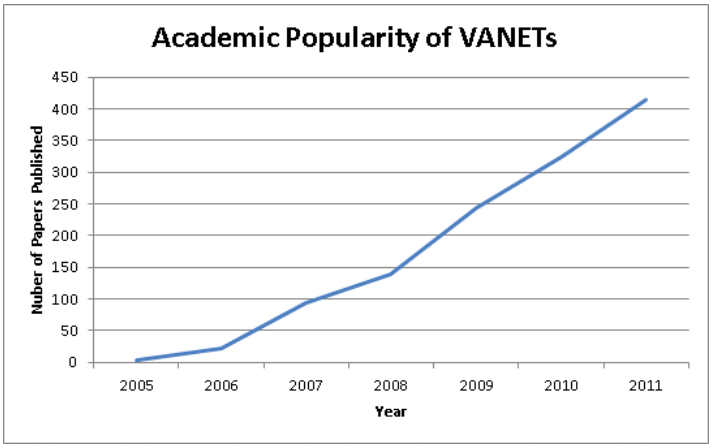

FIGURE 2.1: VANET POPULARITY

Abuelela *et al.* realized that certain VANET applications would benefit from use of the cloud [28]. They proposed using cars as mobile sensor nodes for everything from mapping weather patterns using car's external thermometer, to determining open parking spaces in a parking lot, to intelligent vehicle routing. For large VANETs, a large amount of information could be received at irregular intervals, making a cloud environment the ideal data processing server due to its dynamic resource scaling capabilities. They go further to suggest that vehicles can pool their spare resources in order to form a cloud of their own (imagine how powerful a cloud would be using the spare resources from every vehicle in a mall parking lot).

2.6 ROUTING ALGORITHM

The most popular algorithm used in route planning is A* (A Star) first proposed by Hart *et al*. [29]. A Star is primarily based on Dijkstra's graph search algorithm, developed by computer scientist Edsger Dijkstra. Dijkstra's algorithm functions by always "visiting" the closest intersection. When an intersection is "visited" every connected intersection is added to the "to-be-visited" queue, if it is not already present. The distance to every connected intersection is then updated by determining the sum of the distance between an unvisited intersection and the value of the current intersection; the distance value is then replaced if this value is smaller. The intersection with the smallest distance value in the "to-be-visited" queue is then selected as the next intersection to "visit". The algorithm is usually aborted whenever the destination is "visited"; however, if left unchecked the algorithm will determine the shortest path to every node in the network.

A* improves upon Dijkstra's algorithm by taking into account the distance from the destination in addition to the distance from the source. This focuses the algorithm's search path towards the destination, rather than in a radius as is found in Dijkstra's algorithm. Because of the fact that the distance to the destination is not known (otherwise there would be no need for the algorithm), a heuristic must be used. In the case of vehicle routing, such as that done in this thesis, the L2 distance between the current node and the destination is used.

In the past 10 years with an increase in the popularity of personal GPS route planners (e.g. Garmin devices), researchers have been developing a better cost estimate than distance for vehicle route planning. The most obvious metric, and the one used in this thesis, is mean travel time, however, many additional/supplemental metrics have been proposed. Ambrose *et al.* proposes that not only should travel time be taken into account, but the variance in travel time should be as well [30]. This would aid users who prefer a constant travel speed rather than stop and go traffic. Fleischmann *et al.* realizes that during long trips the current traffic conditions might not be an indicator of conditions an hour from that time [31]. They propose an algorithm which puts more weight in historical data the further from the source the algorithm routes. This would help users avoid congested roads during times of high occupancy, such as rush hour, as well as sections of highway which are prone to traffic jams. Ali *et al.* suggests that weather

forecasts/current conditions be factored into the routing process [32]. Their algorithm would avoid areas prone to flash flooding during rain storms, areas prone to black ice during ice storms, and would advise the user if the conditions are dangerous to drive in along any section of their route. Lastly, Wilkie *et al.* developed a self-aware traffic planner [33]. The planner takes into account other routes it has issued in order to avoid congestion along any one roadway. Unfortunately, in order for this system to be effective a large percentage of vehicles must use the router and all vehicles must follow its recommendations.

2.7 SIMILAR WORK

In the process of conducting a literature review, three papers were discovered which detailed work similar to this thesis. In the first piece, Bessler *et al.* works to route the user towards a charging station within the EVs estimated range which best fits the users vehicle requirements [34]. Part of the algorithm in [34] takes into account if public transportation is available near the station in order to get the user to their destination while the vehicle is charging. Within their simulation, this charging station would then be reserved a single EV. Additionally, the work in [34] presents a heuristic to develop a charging schedule in order to optimize both time between charges and charging station utilization. In the next paper, Worley *et al.* seek to develop a model which would allow for the optimization of the delivery/pickup route of an EV taking into account its limited range and its need for charging [35]. The work in [35] did not simulate or test the model but rely on mathematical proofs.

Lastly, Kobayashi *et al.* provided the inspiration for this thesis [36]. The work in [36] developed an algorithm which produced a route to a given destination based on a pre-calculated remaining distance and distance to the charging station(s). Their algorithm used the estimated distance to retrieve a list from a pre-compiled database of all charging stations in range. They then routed to each of these charge stations using distance as an edge cost and selected the route with the lowest difference between itself and the route directly to the intended destination. Unlike in this thesis, they acknowledged the possibility that a trip may require more than one charge and in this case, they used pre-computed distances between charging stations to develop an optimal-multi charge route.

3 METHODOLOGY

The goal of this thesis was to develop a prototype system which advises the driver of a PHEV/EV of which charging station they should use based on traffic conditions, the amount of power left in the vehicle, and the cost of electricity at the charging stations, representing the state of both the station and the electrical grid. The goal of the prototype system was threefold; to provide a proof-of-concept system to showcase that systems sharing data on the cloud would allow for benefits not achievable otherwise, to provide a system which can be used to analyze behaviors of EVs and the grid, and to provide a system which others can build off of in order to test more focused research.

The system uses the open source program Traveling Salesman as routing engine which uses simulated real-time traffic conditions and current conditions at each charging station to select an optimal charging station for each EV to issue a routing request. The sub systems responsible for reporting traffic conditions and station information are housed in separate Virtual Machines within a small cloud system, in order to emulate the fact that in a real world scenario the routing program would not be able to directly influence these systems. Figure 3.1 illustrates this general architecture. Before the system was designed, the data points which would be taken into consideration and the systems which would supply them were selected. The system takes as input the EV's current location, intended destination, information on the current battery state, and some routing preferences; it returns as output the selected charging station and the route to get there and then proceed on to the destination.

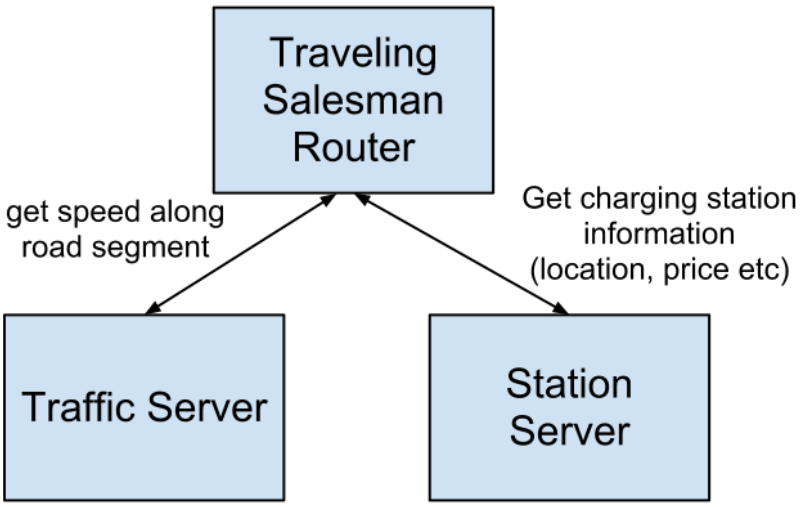

FIGURE 3.1: VIRTUAL MACHINE ARCHITECTURE

3.1 SELECTION OF DATA POINTS AND SOURCES

Position information, individual velocity, and battery status as measured by internal vehicle sensors were selected, as well as average velocity and occupancy along road segments as collected by roadside sensors, and electricity price at charging stations as collected by the smart grid.

3.1.1 VEHICLE DATA

For the purposes of this thesis, information collected through sensors on individual vehicles is assumed to be transmitted through a VANET. Data such as GPS information and average speed would be collected for use by applications such as an ITS and weather mapping and could be shared with an EV routing program; battery charge readings would likely need to be gathered explicitly for this purpose. In order to protect user data, all information pertaining to individual vehicles should be kept anonymous; for example each vehicle should be assigned a UUID and all information should be stored in an encrypted database. This thesis required three data points collected through on-board sensors, GPS location information, average speed, and battery state.

GPS information determines which road segment the car is on. This information can be collected through an internal GPS receiver, a Bluetooth connection to a phone, or triangulated either through a cell signal or through the car's position in the VANET [37].

Average speed was used in estimating the speed along road segments. This information can either be calculated through changes in GPS information or downloaded directly from the accelerometer.

In a fully functional system, information would be collected on the current state of the battery and its rate of depletion in order to estimate the distance a vehicle can travel on the remaining power in its battery. The State of Charge (SoC) is a measure of the remaining charge in the battery. It can be easily calculated using Equation 3.1, where the Current Energy is the current amount of energy in the battery and Max Energy is the amount of power in the battery when it is completely charged.

$$SoC(t) = \frac{Current\ Energy(t)}{Max\ Energy} \tag{3.1}$$

The Energy Index (EI) is a measure of the amount of power expended per unit of distance over a given time interval. This measure is required because the amount of power needed to transverse a given distance changes based on environmental factors such as temperature and traffic, and variance in velocity (stop and go driving). EI is calculated by Equation 3.2 where SoC(t) is the State of Charge at time t, v(s) is the vehicles velocity

at time s, t is the current time, and τ is the amount of time over which this metric measured.

$$EI(t) = \frac{SoC(t-\mathrm{T}) - SoC(t)}{\int_t^{t-\mathrm{T}} v(s)\, ds} \tag{3.2}$$

Using these two metrics Equation 3.3 calculates the estimated remaining distance based on the power remaining in the battery and remaining driving conditions.

$$Estimated\ Distance\ (t) = \frac{SoC(t)}{EI(t)} \tag{3.3}$$

3.1.2 ROADSIDE DATA

Numerous ITS systems already collect useful traffic information such as average speed and occupancy along a road segment. The data collected by ITS systems not only has a greater accuracy; the data also has the benefit of being anonymous by nature. When these ITS applications become hosted within a cloud, this data would be easily shared with other applications within the same data space.

Speed data collected by road-side sensors is more accurate due both to the collection of velocity data from every vehicle, not just those connected to a VANET, and the fact that the data is continuously averaged over a time period, not just the discrete representation collected by a VANET. Two of the primary downsides are that not every road segment has roadside sensor collecting information and that there is a large delay in reporting results.

Occupancy, or the average number of cars on the road segment, is a good measure of the accuracy of the recorded data. For example, a single car moving slowly would not accurately represent the speed along the road segment, since that car would most likely no longer still be present along the segment by the time this data is reported. On the other hand, a large number of cars moving slowly would most likely represent a traffic jam or some other persistent factor. Additionally, with an occupancy of zero one can assume that the speed along a road segment is the speed limit since the road is empty.

3.1.3 CHARGING DATA

Coming up with data to sample from charging stations was one of the most challenging parts of this thesis. The data had to be able to be used to form an optimality metric of both the underling grid and the charging station. Factors, such as whether a

station is full or not and the functionality of the underlying grid, were ignored because no model was readily available to simulate/predict these factors.

The electricity cost was selected as the optimality metric for the smart grid because it represents a station's optimally by its nature. If a station is drawing too much power, or drawing power irregularly, it will pay more for power and this cost will transfer to the user's end cost. By the same token, if a station has become too busy and is running low on reserves the user's end cost will go up. Other factors that affect price, such as competition between vendors and subsidies from large parent companies are assumed to be comparatively negligible [21]. With the introduction of the smart grid, these cost adjustments will happen in real time.

3.2 ARCHITECTURE

The prototype system can be divided into three individual sub systems; the traffic server responsible for the VANET and ITS data, the station server responsible for the smart grid data, and the traveling salesman router which housed the thesis' routing algorithms. All three subsystems were encapsulated within individual virtual machines within a small cloud system.

3.2.1 VIRTUAL MACHINES

The processing power and algorithm used to select the optimal station will be contained within the Traveling Salesman Routing Program. In a real world scenario the administrators of this program would not have direct access to the real-world data the algorithm would require. They would have to go through third parties in order to gain access to real-time smart grid and traffic data. The simulation emulates this by encapsulating these functionalities within their own VMs running on the cloud. The programs communicate via Java's built-in Remote Method Invocation (RMI) protocol [38]. The Traveling Salesman Program functions as an RMI client, and both the Traffic Server and Station Server function as RMI servers.

The basic structure of an RMI-based method call involves a client, a server, and a registry. To make a call to a remote object, the client first looks up the remote object on which it wishes to invoke a method in the registry. The registry returns a reference to this server hosted object, which the client can use to invoke any methods that the remote object implements. The client communicates with the remote object via a user-defined interface which is actually implemented by the remote object. The client does not deal directly with the remote object at all, but with a code stub (reference) which deals with the process of communication between client and server (using sockets in most cases). Similarly, the remote object does not directly process these calls, but rather a code skeleton handles communication with the code stub and relays all method calls, depicted in Figure 3.2. This allows both the client and server to function as if a local object was being manipulated, abstracting away communication details.

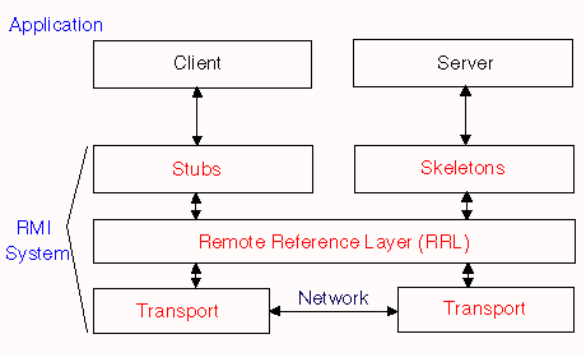

FIGURE 3.2: RMI CALL DIAGRAM

3.2.2 STATION SERVER

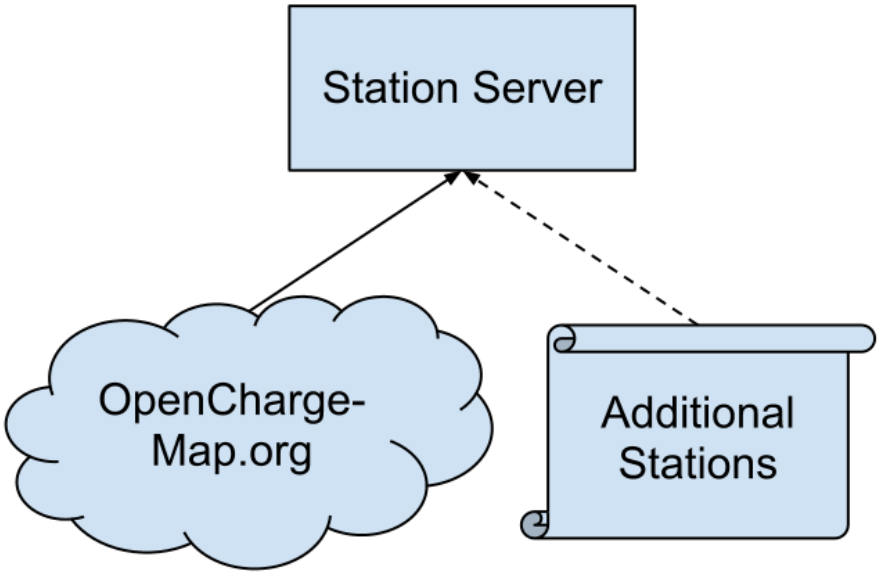

FIGURE 3.3: STATION SERVER ARCHITECTURE

The charging station server emulates smart grid data as closely as possible by querying real charging station locations and types from OpenChargeMap.org. An electricity price is then artificially generated for each station by introducing a random deviation from a set base price. For the purposes of this system, the cost was generated based on the current cost of electricity from Rochester Gas and Electric and a 10% standard deviation. Additional functionality was added to manually introduce charging stations to the server. This functionality allowed for the exploration of various "what if" scenarios such as: "If a station is added at a chosen locationu76 what will happen to the success rate and load on the other stations?" In an ideal system the server would include models for the charging station and underlying electrical grid, however in this prototype system these were left out as they were not practical for use in this thesis. Without these

models, the duration of a charge could not be simulated due to the fact the station output is unknown; without charge duration, station occupancy could not be simulated because it is unknown how long an EV will occupy a station. Further, without both models the load on the electrical grid could not be simulated and thus grid optimality could not be directly determined, so as discussed in section 3.1.3 electricity cost is used.

3.2.3 TRAFFIC SERVER

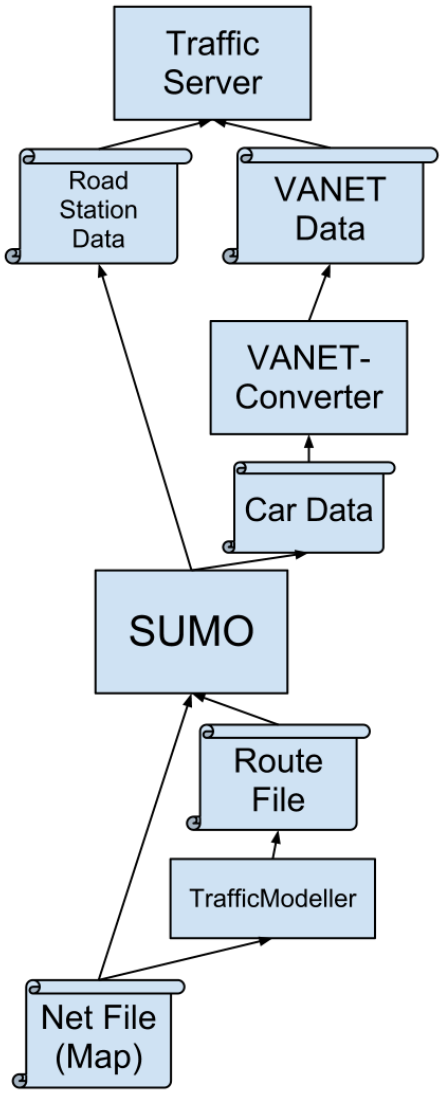

FIGURE 3.4: TRAFFIC SERVER ARCHITECTURE

Before the simulation proper starts, a traffic simulation must first be run in order to populate the ITS and VANET data. This is done through the SUMO traffic simulator. After the simulation is run the VANET converter emulates the data that would be generated by a VANET. The traffic server then blends both ITS data and VANET data in

increments as the simulation is advanced. The server also processes queries from the router for the predicted speed along a given road segment.

3.2.3.1 SUMO

The SUMO simulator takes two main input files, a net file, converted from the open street map file, and a route file, depicting cars and the paths they are to take. The net file has approximately the same road segments as the osm file, however due to the conversion process they are not identical, a phenomenon similar to what would be found under real world operating conditions. The traffic server must account for these inconsistencies. Generating route files posed a challenge, functionality was included with sumo to generate random routes; however this would not have been representative of a real-world environment. It was desired to get as close to realistic traffic conditions as possible. Eventually TrafficModeller was discovered, the program uses population data, job distributions, school locations and other commonly available regional statistics to generate realistic traffic for SUMO.

SUMO includes functionality to simulate an ITS traffic sensor which averages car speeds over road segments during a given sample period; it then outputs this data into a road network "dump" (file). It also includes functionality for each individual vehicle to report its current speed and location every so often. This would represent an ideal VANET, however the most realistic test data is needed so a script was written in order to produce a data file which introduces an artificial delay based on the location of generated access points and only includes a fraction of the cars from the original "dump".

3.2.3.2 VANET SCRIPT

SUMO also has the ability to dump every car's position and average speed after a configurable time interval. This car dump was then run through a VANET converter script, whose process flow chart is included as Figure 3.5, in order to emulate the data that would be generated by a VANET by the introduction of a realistic time delay.

This script either randomly or methodically determines the location of roadside base stations, and then artificially introduces time delays based on each vehicle's distance from its closest base station. In grid mode, access point locations are placed in a grid pattern separated by a user specified interval. The upper left corner is the first AP placed, and thus is the only AP found across all grid-generated patterns. In random mode, a user specified number of access points is randomly placed within the bounds of the supplied SUMO map. Access points can be saved for re-use.

The car dump file is then loaded one time interval at a time. Each car element is processed individually. A dictionary is kept which is comprised of vehicle IDs and whether they are included in the VANET. The first time a vehicle is encountered it is determined if it is included in the VANET. This is done by randomly generating a value and comparing it to a user defined penetration level. For a sufficiently large number of vehicles, this penetration ratio will be equivalent to the ratio of cars in the network as compared to the original dump file. A Boolean value representing this result is recorded

in the dictionary for the next time the vehicle is encountered. If the vehicle is in the network, a champion algorithm determines the smallest L2 distance to an Access Point. This distance is then used to estimate delivery delay by using a formula created by interpolating the results published by Chuah *et al.* [26]. This delay is then used to calculate the packet's delivery time before the element is re-written to a VANET file.

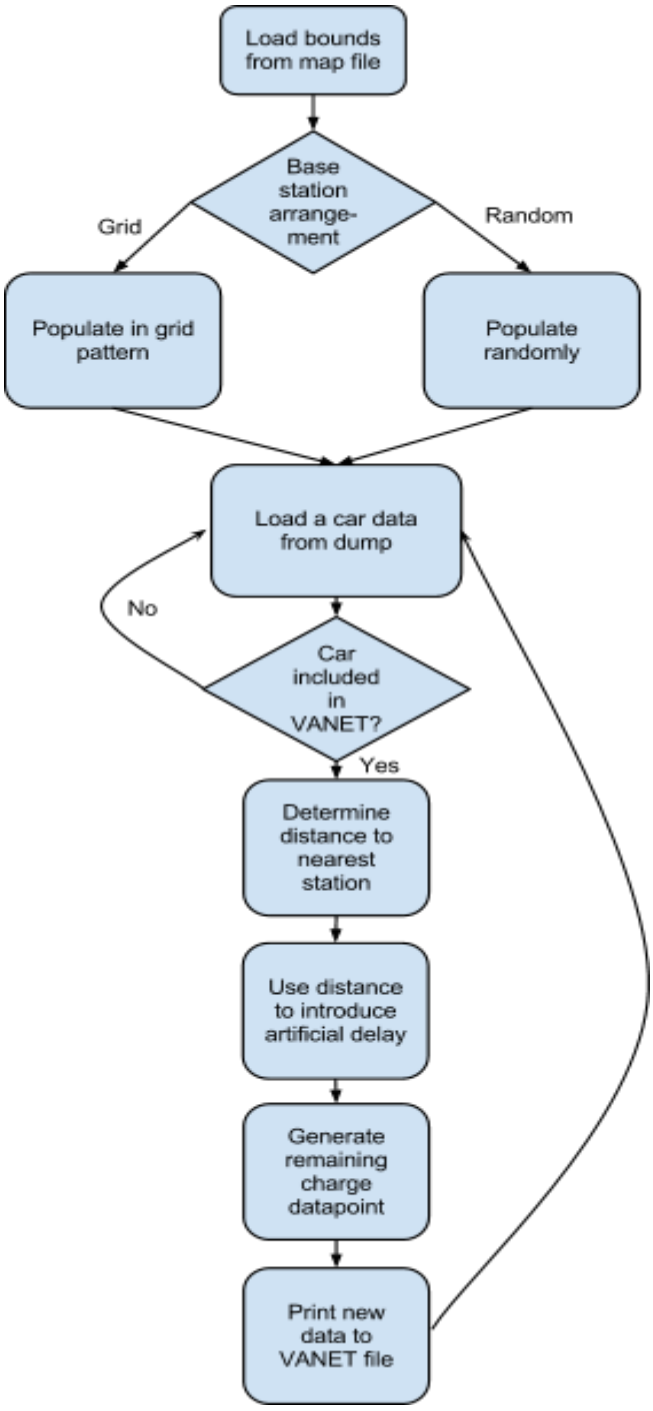

FIGURE 3.5: VANET CONVERTER SCRIPT FLOW DIAGRAM

3.2.3.3 TRAFFIC DATA FUSION

Because VANET data only represents a brief snapshot of the vehicle, and not all vehicles contribute VANET data, ITS data is considered more reliable. However, its information is reported less frequently, sometimes as slow as every couple of hours. An algorithm was developed to blend the more reliable ITS data with the more available VANET data. Because ITS data is more reliable, VANET data is used only to fill the gap until the next set of ITS data arrives. The average velocity is calculated via an exponential moving average, which gives more weight to the more recent data (the most recent measurements are a better predictor of the current speed). This formula is included as Formula 3.4

$$V_{avg} = \alpha \cdot V_{ITS} + (1 - \alpha) \cdot V_{avg} \tag{3.4}$$

Because ITS data has such a large update time, VANET data is used as a supplement until the next batch of ITS data arrives. First, a running average of speeds reported from each car is kept. Next, these car averages are averaged to obtain $V_{VANET,}$ the current average along the road segment. The predicted speed, V'_{avg}, is then calculated via Formula 3.5. Because the VANET data does not cover the same time span as an ITS update, it's α value must be scaled appropriately. Formula 3.6 accomplishes this task; please note that T is the length of an ITS update interval, and T' is the length between the latest ITS update and the latest VANET update.

$$V'_{avg} = \alpha' \cdot V_{VANET} + (1 - \alpha') \cdot V_{avg} \tag{3.5}$$

$$\alpha' = \alpha \cdot \frac{T'}{T + T'} \tag{3.6}$$

3.2.4 MODIFIED TRAVELING SALESMAN

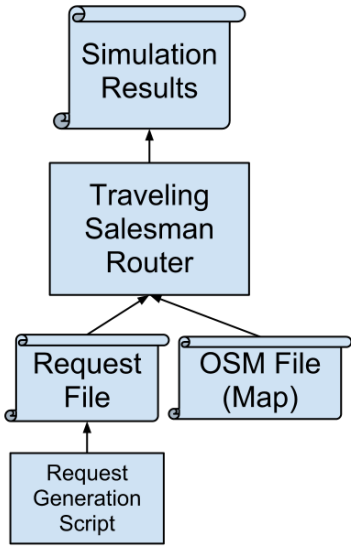

FIGURE 3.6: MODIFIED TRAVELING SALESMAN ARCHITECTURE

This is the primary algorithm of the thesis. The modified traveling salesman algorithm takes in two input files, an OpenStreetMap data file (NOTE: this must correspond to the network file used with the SUMO simulator), and a file listing the requests to be simulated. The program then outputs the results in an easy to read xml format. This architecture is depicted in Figure 3.6.

3.2.4.1 REQUEST SCRIPT

A script was created to randomly generate routing requests in order to eliminate the complications which would arise from synchronization with the traffic simulator. When the script is started the user specifies a simulation time range, a map file, and the number of requests to generate. Generated requests include the vehicle's current location, its intended destination, the amount of battery it has left, a metric estimating the distance to energy usage ratio for current road conditions, the level of the charge station to search for, and the time the request was received. Both the current location and destination are randomly distributed uniformly throughout the area of the given map file. The amount of battery the vehicle has remaining, also known as its State of Charge (SOC), is determined as a function of the absolute distance between the current location and destination. The Energy Index, or rather the estimation of the SOC depletion per meter, is generated uniformly between a pre-selected range [16]. This range was determined by a general survey of road tests on popular electric vehicles that provided max, min and average travel ranges for a fully charged vehicle. It is ideal that most requests should be successful without being able to reach the destination. In order to accomplish this, the fact that the shortest distance between two points is a straight line was utilized, as well as the fact that there is rarely a straight route between two points on a road network. The L2 distance between the current location and destination is calculated, and the SOC is set using the EI so that the estimated distance function produces this value. During testing, the charge station level was always two, as that is the only type currently available in Rochester. Lastly, the received time is randomly distributed over the simulation start and stop times, specified by the user.

3.3 ELECTRIC VEHICLE ROUTER

Rather than reinventing the wheel and developing a vehicle routing program when many already exist, the open source program named “Traveling Salesman” was selected for use in the experiment [39]. It was selected based on its modular design and its base language of Java, which behaves the same across all operating systems and platforms. The Traveling Salesman program uses information from OpenStreetMap (a street map database) and routs vehicles to their destination based on road lengths. It preforms routing using a modified version of the A* algorithm, which has been optimized to incorporate two-way roads and turn restrictions [39]. In the original program the graph edges represent the length of the corresponding road segment. However, the electric vehicle router, developed as part of this thesis, modified the weights of the edges of the network based on predicted traffic density and travel speed. Possible charge stations are provided to the program based on the location of the vehicle and its estimated travel

radius. The final destination selection then balanced the travel time and convenience for the vehicle operator with the constraints of the distribution grid.

3.3.1 TRAVELING SALESMAN FRAMEWORK

The Traveling Salesman program is extremely modular, and therefore required only a few modifications in order to use it for the purpose of this thesis. When evaluating the cost to traverse a road segment, the router module asks the selector module if the segment is valid (e.g. a car cannot traverse a bike trail), then asks a metric module what the cost of traversing said segment is. The metric module then calculates the cost and returns it to the router, eliminating the need for any changes to be made in the router module, or any other module, whenever a different evaluation method is desired. This cumulates to the fact that only a couple modifications were made; the required external data was be loaded into the program, a heuristic was used to predict the range of the vehicle in question, a metric class was created which evaluates the cost of traversing a road segment based on the estimated current traffic speed along that segment. This entire process is depicted in Figure 3.7.

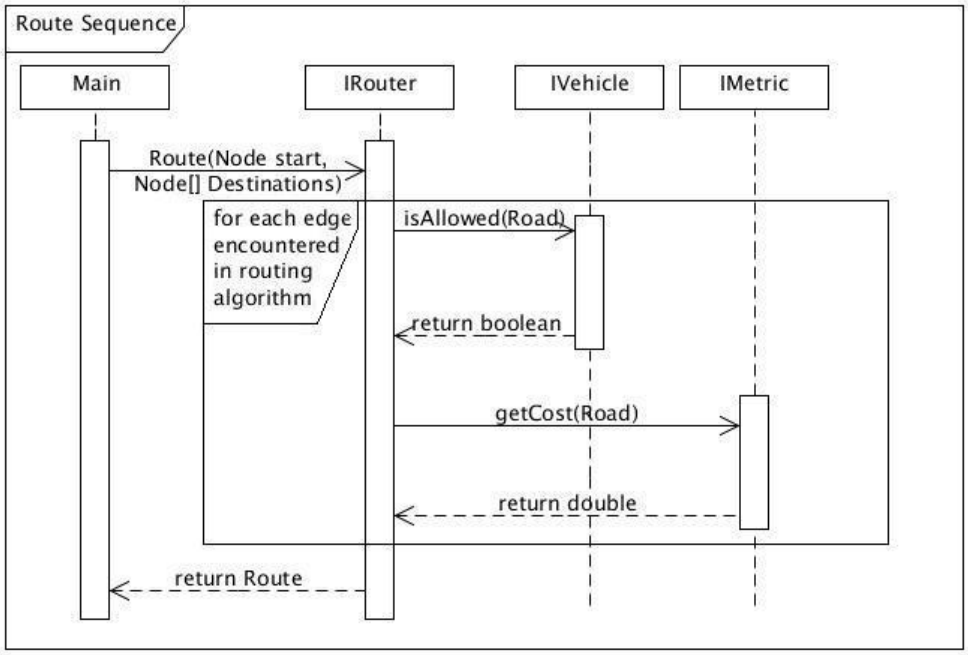

FIGURE 3.7:TRAVELING SALESMAN SEQUENCE DIAGRAM

3.3.2 FASTEST ROUTE METRIC

```
1  FastestRouteMetric
2
3  Private HashMap<String,Double> memory=new HashMap()
4
5  double getCost(aSegment):
6     String segID=convertToSumoID(aSegment)
7     if(memory.contains(segID){
8           double time= memory.get(segID)
9     }
10    else{
11          double length = aSegment.getLength()
12          double speed=TrafficServer.getSpeed(segID)
13          if(!isValidSpeed(speed){
14                speed = getSpeedLimit(aSegment)
15          }
16          double time=length/speed
17          memory.put(segID,time)
18    }
19
20    aSegment.setCost(time)
21    return time
```

The metric created for use in the prototype system calculated the estimated amount of time it would take a car to transverse a road segment based on the length of the segment and the average speed of the vehicles traveling along the segment as retrieved from the traffic server. The pseudo code for this metric is included above. The traffic server stores its information using SUMO IDs, which differentiate different directions along the road segment by different signage (a road going from west to east is positive, east to west is negative), so the first step of determining the cost is converting the segmentID to its SUMO equivalent (line 6).

In order to reduce both communication costs and computation time, the results of recent road segment evaluations are stored in a static hashmap. The next step of the algorithm is to check to see if this collection contains the travel time for the current segment (lines 7-9), eliminating the repetition of already performed work. If the travel time does need to be calculated the length of the segment is retrieved (line 11), which has been stored in the object's metadata, and the traffic server is queried for the segments speed (line12). If the server has no speed estimate for the requested road segment, it returns an error value and the metric assumes that you may travel the speed limit (lines 13-15). Once both the segment's length and speed are retrieved ,the estimated travel time is calculated (line 16) and stored in the hashmap in case the segment is encountered in the near future (line 17). Lastly, the cost is stored in the segments metadata for future use (line 20) before cost is returned to the calling function (line 21).

3.3.3 MODIFIED A* ALGORITHM

Optimizations were made to the A* routing algorithm in order to take advantage of the fact that the algorithm calculates the shortest distance between the source and every node it encounters. Because the prototype system routes from a single source to multiple charging stations and from these charging stations to a single destination, re-calculating the Dijkstra maps each time would be a waste. Instead, the program re-uses these maps when plotting between the source and every charging station. The program then uses a new map and plots between the destination and every station. This is possible because the road map is stored as an undirected map; the directional information for the road segment is stored as metadata. When routing backwards, it is essential to make sure that the road segment is still evaluated in the correct direction. The second half of the route is then reversed (so that it is going from the station to the destination), before both halves are combined into the final route. The increase in efficiency is slightly reduced by the fact that the map must be checked if the end node has already been processed, and the to-be-visited queue must be re-ordered using both the A* heuristic and the new end node.

Additionally, functionality was introduced so that the router did not exceed a set distance from the start node. This ensured that all routes between the source and charging stations are less than the vehicle's estimated travel distance and that the router does not waste time with impossibly long routes. Initial testing showed an average decrease of 42% in processing time with both enhancements in place.

3.3.4 ELECTRIC VEHICLE ROUTING ALGORITHM

```
1  Main Routing Program
2
3  StationContainer getRoute(source, destination, EI, SOC,
      stationType, optimalityMetrics):
4
5     Node startNode=map.findClosestNode(source)
6     Node endNode=map.findClostestNode(destination)
7     double estimatedDistance=SOC/EI
8     aStar.resetMemory()
9     Route masterRoute=aStar.routeWithMemory(source,destination)
10    if(masterRoute.Distance<estimatedDistance){
11          return new Station("charge not needed")
12    }
13
14    Stations[]=stationServer.getStations(stationType,
                        estimatedDistance)
15    route[] firstHalf=new array
16    route[] secondHalf=new array
17    For(currentStation in Stations){
18          firstHalf[i]=aStar.routeWithMemory(startNode,
                              currentStation, estimatedDistance)
19          if(firstHalf[i]==null){
```

```
20                Stations.remove(currentStation)
21          }else if(firstHalf[i].Distance<estimatedDistance){
22                Stations.remove(currentStation)
23          }
24    }
25
26    aStar.resetMemory()
27
28    For(currentStation in Stations){
29          secondHalf[i]=aStar.routeWithMemory(destinationNode,
                                                currentStation)
30          if(secondHalf[i]==null){
31                Stations.remove(currentStation)
32          }
33
34    completeRoute=firstHalf[i]+reverseRoute(secondHalf[i])
35          costDiff= completeRoute.Distance-masterRoute.Distance
36          currentStation.setRoute(completeRoute, costDiff)
37    }
38
39    bestStation= findOptimumStation(Stations
                                      [],optimalityMetrics)
40    if(bestStation==null){
41          return new Station("no stations in range")
42    }else{
43          return bestStation
44    }
45
46  Station findOptimumStation(Stations [],optimalityMetrics):
47    if(StationContainers.size < 1){
48          return null
49    }else if(StationContainers.size == 1){
50          StationContainers[0].setFavioritability(1)
51          return StationContainers[0]
52    }else{
53          double maxCost,maxRoute = -∞
54          double minCost,minRoute = ∞
55
56          for (currentStation in Stations){
57                if(currentStation.getRouteCost > maxRoute){
58                      maxRoute = currentStation.RouteCost
59                }
60                if(currentStation.getRouteCost < minRoute){
61                      minRoute = currentStation.RouteCost
62                }
63                if(currentStation.getDistance > maxCost){
64                      maxCost = currentStation.ElectricityCost
65                }
66                if(currentStation.getDistance < minCost){
```

```
67                      minCost = currentStation.ElectricityCost
68                }
69          }
70          Station BestStation=null;
71          double bestValue=∞
72          for (currentStation in Stations){
73                normalRoute= (currentStation.RouteCost-
                              minRoute)/(maxRoute-MinRoute)
74                normalCost= (currentStation.ElectricityCost –
                              minCost)/(maxCost-MinCost)
75                value= optimalityMetrics[0]*normalRoute +
                        optimalityMetrics[1]*normalCost
76                currentStation.Optimality = value
77                if(bestValue<value){
78                      bestValue=value
79                      bestStation=currentStation
80                }
81          }
82          return bestStation
83    }
```

This sub-section discusses the primary class for the prototype system. It contains two methods of note: the routing method, and the method that contains the algorithm to determine the optimal charging station. The pseudo code for both sections are included above, however it should be noted that simulation functionality, such as advancing simulation time and calculating delays was left out as it would not be implemented in a real system.

The route method is the driving force of the prototype system. First, three important variable must be calculated, the node closest to the vehicles current position (line 5), the node closest to the vehicle's destination (line6), and the estimated distance the vehicle can travel without charging (line 7). Next, a route between the car's position and destination is calculated, both to ensure that the vehicle does indeed need a charge to reach its destination and to use as a comparison to determine how far a charging station is out of the way (lines 9-12). A list of charging stations of the given type within a radius of the estimated distance from the EV's location is retrieved from the station server (line 14). The program then begins routing to each of these servers (lines 15-37). The first half of the route (destination to station) is calculated for each station in order to take advantage of the optimizations discussed earlier in Section 3.3.3. Because all of the retrieved stations are within a radius from the EV's location, the actual route to the station may still be longer than the estimated remaining distance; in such a case, the station is removed from the list of candidates (lines 19-23). The A* router keeps track of the Dijkstra map so the memory must be explicitly reset every time a new source node is used (line 8, line 26). Next, the second half of the route is found for each charging station (lines 29-32), because due to the optimizations, these routes are calculated backwards (destination to station), they must be reversed before being combined with the

first half of the route and then stored as metadata in the station object (line 34-36). Finally, the list of stations is sent to the optimality algorithm (line 39) to determine the best station which is then returned to the calling method. The route method is visualized in Figure 3.8

The optimality algorithm is operates as follows. First, the boundary conditions are evaluated. If the given collection of stations is empty an error value, in this case null, is returned (line 48). If there is only one station in the collection then it is selected as the best choice (line 51). For more than one station, a champion algorithm is performed to get the minimum and maximum of all values which are used to determine the optimum station, in this case electricity costs, and the difference in path costs (lines 53-69). These min/max values are then used to normalize the values to between the zero and one (lines 73-74) for each station. The normalized costs are then scaled according to the supplied preferences (line 75), and another champion algorithm is performed, in order determine the most optimum station (lower is better).

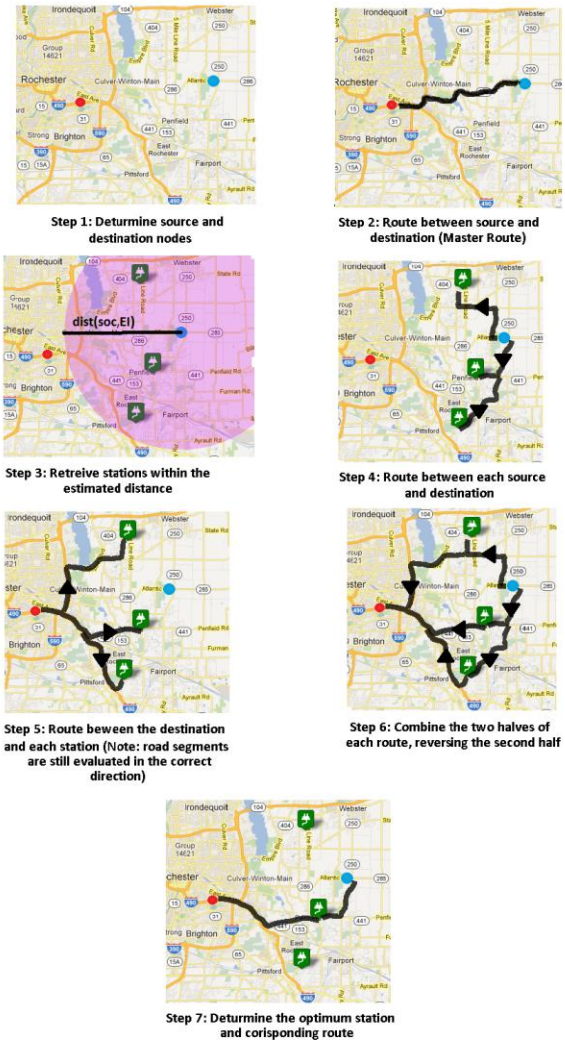

FIGURE 3.8: STEPS OF THE ELECTRIC VEHICLE ROUTING ALGOITHM

3.4 CLIENT PROTOTYPE

In order to demonstrate this system's use, a prototype client Graphical User Interface (GUI) was created. This program resided on a non-cloud machine while communicating with the subsystems residing within a cloud environment. When the program first starts, the user is presented with a screen requesting the IP address of the primary routing server and the local location of the OSM file used for routing. This screen is included as Figure 3.9.

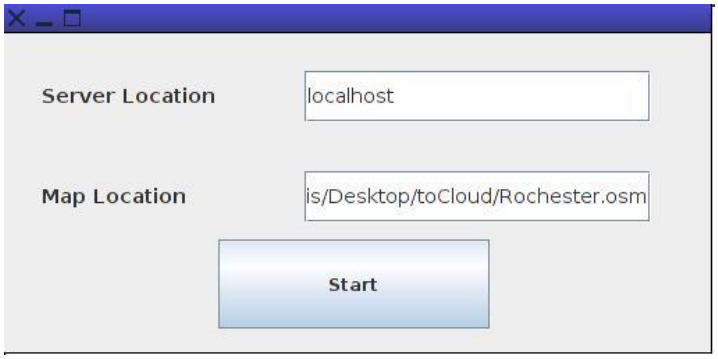

FIGURE 3.9: CLIENT SETTINGS SCREEN

Next, the user is presented with a screen requesting routing parameters. These parameters are the same as those included in a routing request discussed in Section 3.2.4.1. As a bonus, the program calculated the estimated distance the vehicle can travel based on the SOC and EI and displays this distance to the user. This screen is included as Figure 3.10 .

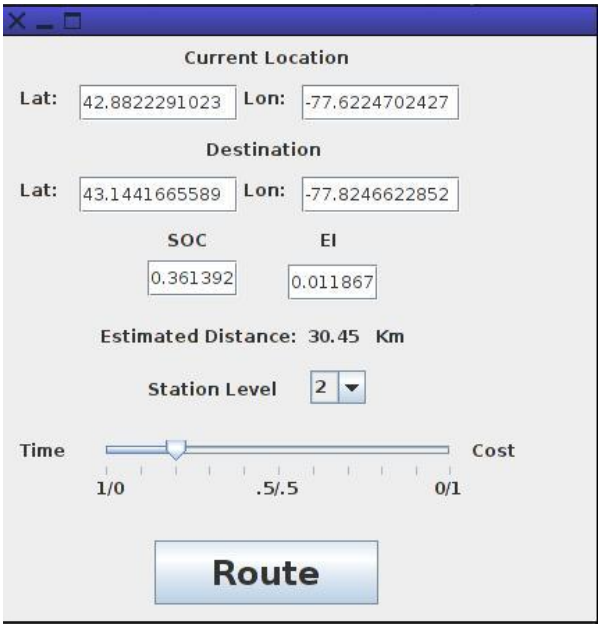

FIGURE 3.10: CLIENT PARAMETERS SCREEN

The program then sends these parameters to the primary routing server as a routing request. The server returns both the selected station and the optimal detour route. This information is displayed utilizing functionality taken from the Traveling Salesman Routing Program. A sample map is included as Figure 3.11. The green dot denoted the location of the EV, the red dot denotes the destination, and the yellow dot denotes the charging station location. The detour cost difference and price of electricity are displayed for user connivance.

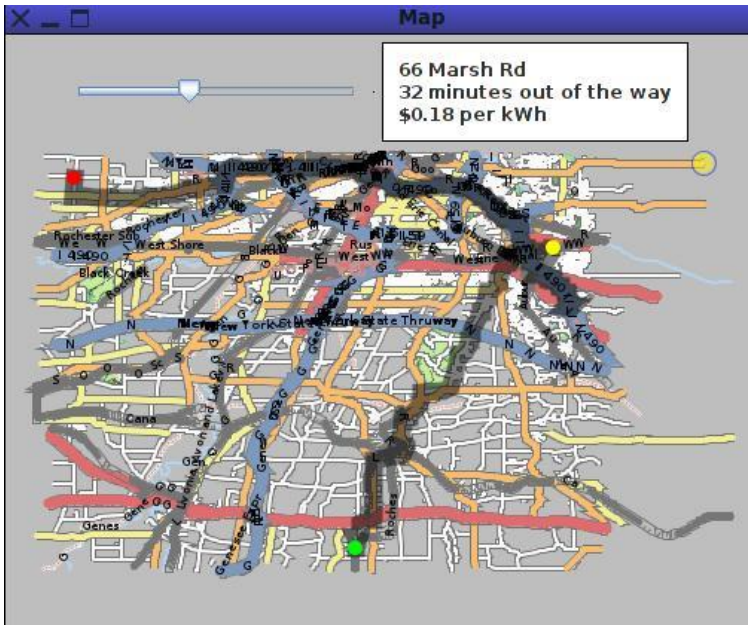

FIGURE 3.11: CLIENT MAP SCREEN

3.5 INCORPORATION OF ADDITIONAL DATA POINTS

More data points used to select a charging station should be incorporated in the findBestStation method; such factors include the time it would take to charge at a station. These factors should be normalized across all charging stations, scaled according to a preference factor (such as alpha and beta already used in this method) and added to the optimality metric. Note that the optimality factors should add to 1 so that the optimality metric itself is always between 0 and 1. If possible, these additional factors should be stored in the station's metadata so that the number of parameters passed to the method is kept to a minimum.

If additional factors are being added to route selection, these factors should be either added to the FastestRouteMetric class's getCost function or to the getCost function of a new metric. This factor should be able to be determined for each road segment and can either be stored in the segment object's metadata at run time or retrieved from an external object when the function is called. It should be noted that processing road segments require quite a few calls, so if the data is not subject to change it should be stored in a fast retrieval database, such as a hash map.

4 EXPERIMENTS AND DISCUSSION

The prototype system was designed to accomplish the following goals: to provide a proof-of-concept system to showcase that systems sharing data on the cloud would allow for benefits not achievable otherwise, to provide a system which can be used to analyze behaviors of EVs and the grid, and to provide a system which others can build off of in order to test more focused research. The completion of these goals was demonstrated through a series of simulations and a case study using data from the system.

All simulations were conducted on a small three machine cloud running Cent-OS 5.6 and Eucalyptus 2.0.3 cloud software. Each machine had 6GB Random Access Memory (RAM) and ran on an Intel i5-750 which has four cores that run at 2.67 MHz. Each of the three VMs used in the tests were allocated using the specs included as Table 4.1.

TABLE 4.1

VM SPECS

CPU	Intel i5-750 2 x 2.67 GHz
Bus frequency	1.33 GHz
RAM	2 GB
Total Address Space	10 GB

For the purpose of this thesis, all experiments were conducted on a map obtained from OpenStreetMap.org, incorporating the Rochester NY USA area (42.867°,-77.855° by 43.166°,-77.353° which is 40.82 km by 23.22 km). This area is visualized in Figure 4.1. The prototype system processed a series of requests each consisting of a source location, destination location, battery information which was used to determine a vehicle‘s estimated remaining distance, and an indication of whether to give more preference to additional travel time or cost of electricity at the charging stations. The generation of these requests is discussed in section 3.2.4.1. Table 4.2 indicates typical EV full-battery ranges on a fully charged battery and their associated Energy Index (indicator of driving conditions). Unless otherwise noted the simulations consisted of 150 requests generated over the course of an hour. The origin and destinations of these requests are randomly placed within the bounds of the map. Only type 2 stations are considered, they are the only stations present within the map bounds.

TABLE 4.2

TYPICAL EV RANGE

Conditions	Range (km)	EI
Highway	150	.0066
Average	95	.0105
City	65	.0153

Figure 4.1 depicts the locations of the charging stations stored by OpenChargeMap within the map area as well as some sample routes. The system chooses a charging station based on the price of electricity at the charging stations as well as the additional travel time to reach the stations. This additional time is calculated by the difference in travel time of the optimal direct route and the route which includes the station.

FIGURE 4.1: SAMPLE ROUTES

4.1 PARAMETERS

System parameters had to be carefully selected in order to produce realistic results. The parameters which remained constant throughout all simulations are discussed below.

4.1.1 SELECTED BY LITERATURE REVIEW

Instead of preforming detailed tests on every input parameter, those depicted in Table 4.3 were selected based on information found during the literature review. VANET penetration, or the fraction of cars with VANET capability, was selected to match that used by Chuah *et al.* whose data was used to determine VANET packet delay [26]. Both ITS and VANET update frequencies were selected based on the realistic times detailed by Mimbela *et al.* [23]. ITS data updates in 15, 30, or 60 minute intervals; a 15 minute interval was selected in order to give a sufficient number of updates during a typical one hour test. VANET data updates in 2-5 minute intervals; a 4 minute interval was selected as an update frequency In order to give VANET packets sufficient time to reach their destination.

TABLE 4.3

LIT REVIEW PARAMETERS

VANET Penetration	.33
ITS Update	15 minutes
VANET Update	4 minutes

4.1.2 AP DISTANCE VS. AVG DELAY

In order to determine separation between access points placed in a grid pattern, the VANET script was run with a variety of distances between Access Points (APs); the average packet delay and the total number APs were then plotted. As expected the average packet delay increased at a constant rate, as shown in Figure 4.3, and the number of APs decreased polynomial at a rate of x^{-2}, as shown in Figure 4.4. A distance of two and a half miles was selected due to the fact that average delay of a little over a second showed that the prototype system could handle a variance in delivery time while requiring a reasonably small number of APs to implement.

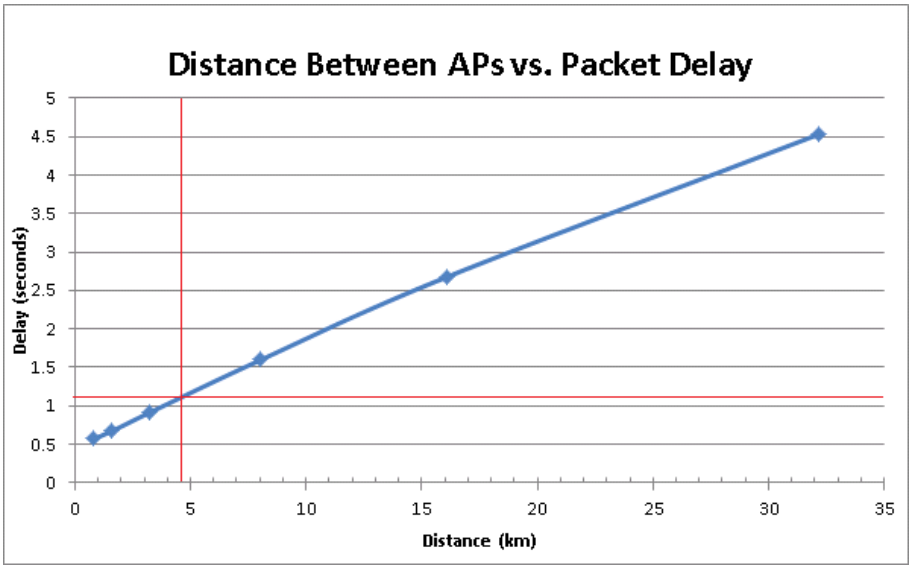

FIGURE 4.2: AP DISTANCE VS. PACKET DELAY

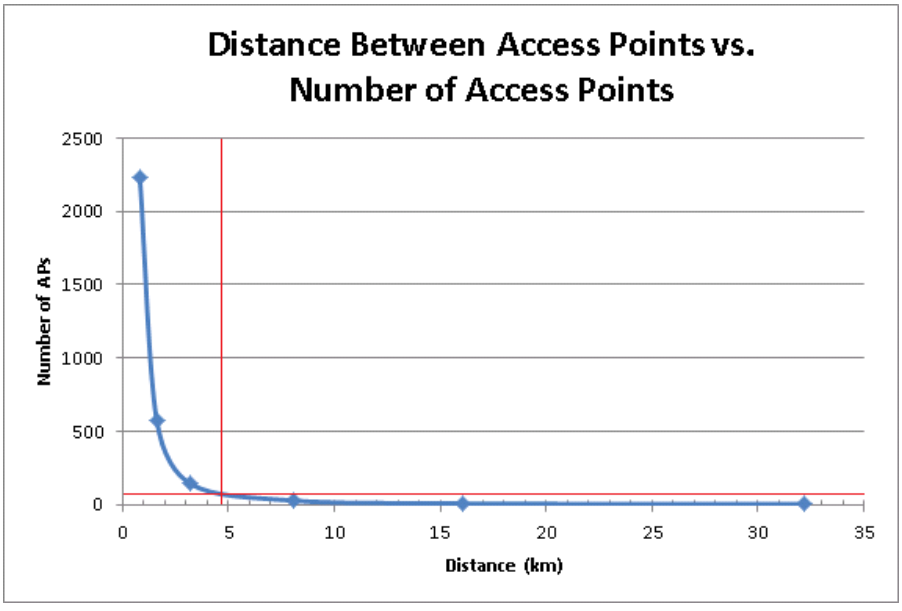

FIGURE 4.3: AP DISATANCE VS. NUMBER OF APS

4.1.3 VALIDITY OF SOC SELECTION

When generation routing requests only certain State of Charges (SOC) were considered, tests were performed to ensure that these assumptions were both valid and produced good results. In order to reduce the number of "charge not needed" results, the estimated distance was calculated as the L2 distance between the EV's current position and its destination. The SOC was then reverse calculated using the estimated distance and the already calculated EI. Because the EV must travel in a straight line in order reach its destination, this reduced the number of "charge not needed" results while maximizing the number of charging stations reachable by the EV. Additionally, no SOCs less than .1 (10%) were considered as it was assumed that a driver would not let his battery get that low without a plan to charge.

One thousand requests were generated and plotted in the histogram included as Figure 4.4. As expected the distances fell in a normal distribution guaranteeing a full range of test distances while thoroughly testing the more typical travel distances. For convenience the horizontal distance and diagonal distance for the map of Rochester used in simulations are marked in the figure.

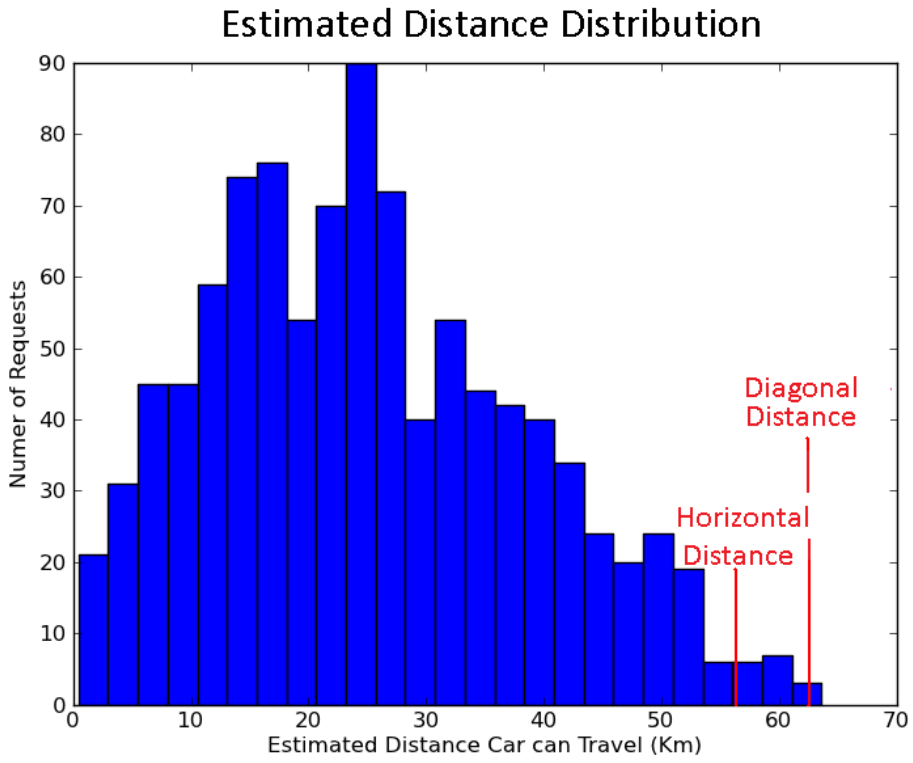

FIGURE 4.4: TYPICAL ESTIMATED DISTANCE DISTRIBUTION

These requests were then simulated under the rush hour conditions described in the next section. Figure 4.5 provides a breakdown of the effectiveness of the simulation. As shown in this figure, a vast majority of the requests successfully produced a

recommended charging station while failing to produce a station an adequate number of times, allowing for improvement in the case study described in 4.2.3. Requests which did not need a charge station either had a straight path between source and destination or had a slight inconsistency in road segment lengths.

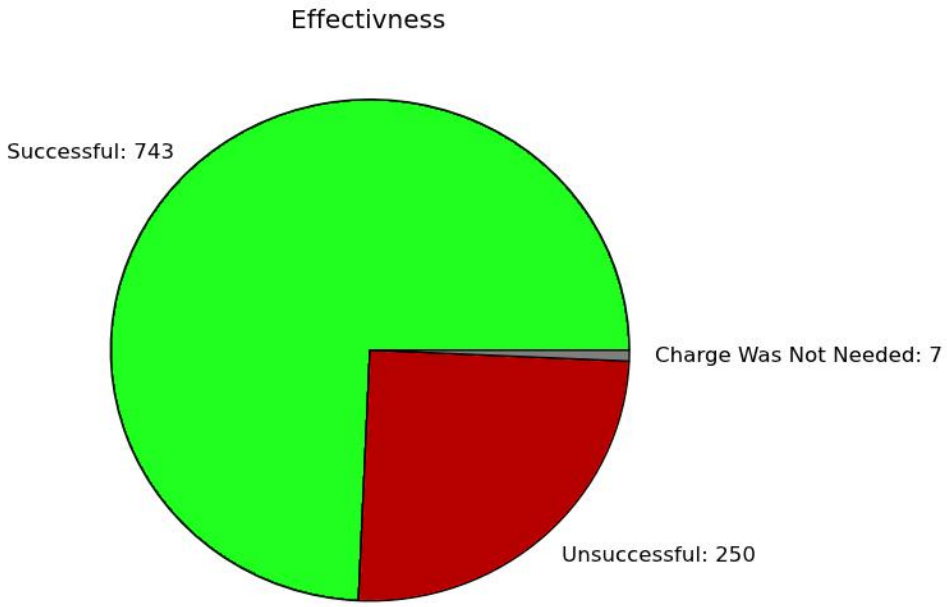

FIGURE 4.5: TYPICAL SUCCESS RATE

4.2 RESULTS

4.2.1 PERFORMANCE VS. ESTIMATED DISTANCE

First, system performance is evaluated and it is determined how the performance scales with when compared to an EV's estimated remaining distance. One thousand random requests were generated under the rush hour conditions outlined in the next section. These requests were simulated and computation time, RAM usage, and CPU utilization were recorded. For reference, the specifications for a Garmin Nuvi 255w, an entry level GPS routing device, are included as Table 4.4.

TABLE 4.4

GPS ROUTER TECHNICAL SPECIFICATIONS [37]

CPU	ARM926 333-MHz
Bus Frequency	166 MHz
RAM	.128 MB
Total Address Space	4 GB

4.2.1.1 RAM

Figure 4.6 plots the maximum amount of Random Access Memory (RAM) used for each request verses the L2 distance between the request's source and destination nodes. Within the system the largest sources of RAM usage are the collection objects used for the Dijkstra map within the A* router. When Java collections are close to running out of space they allocate another large chunk of memory. This allocation method is responsible for the line clusters visible in the plot. As demonstrated by this plot, there is no correlation between the amount of RAM utilized and the request's distance. It should be noted however that the mean memory usage of 805 MB is 6,290 times greater than the amount of memory found in a conventional GPS router.

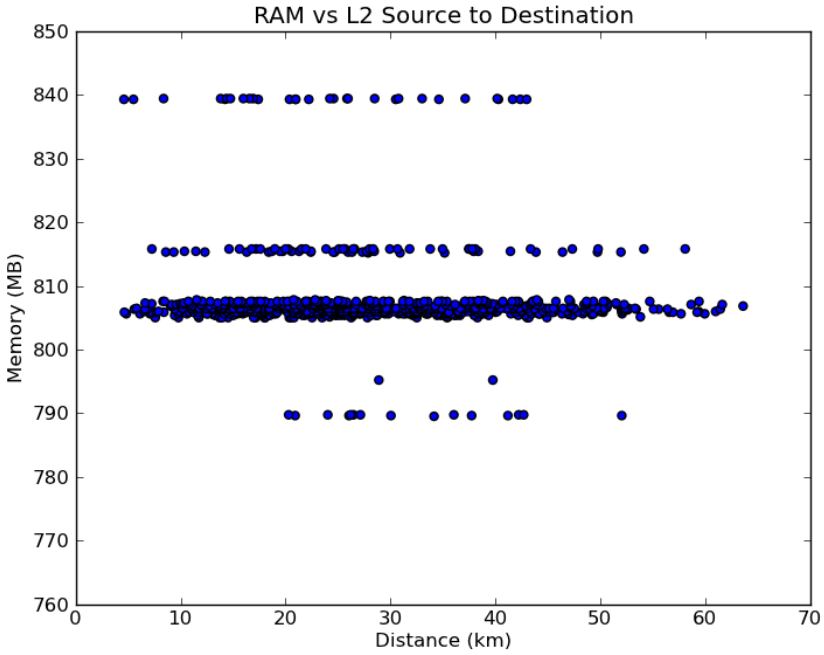

FIGURE 4.6: MEMORY USAGE VS. REQUEST DISTACNE

4.2.1.2 CPU

The CPU utilization appeared to grow logarithmically when plotted versus the request distance, included as Figure 4.7. In order to verify this, a least squares fit was performed and produced Equation 4.1. It was then determined that 78% of the data fell

within 1 standard deviation, 97.57% fell within 2 standard deviations, and all the data fell within 3 standard deviations of this fit, classifying this equation as a good fit. Because of the relatively small amount of processing power being utilized, the system could be implemented with significantly less powerful processors. It should be noted however that the processor indicated in Table 4.4 is 12.5% as powerful as the processor used in this simulation and would only be powerful enough for a small handful of cases.

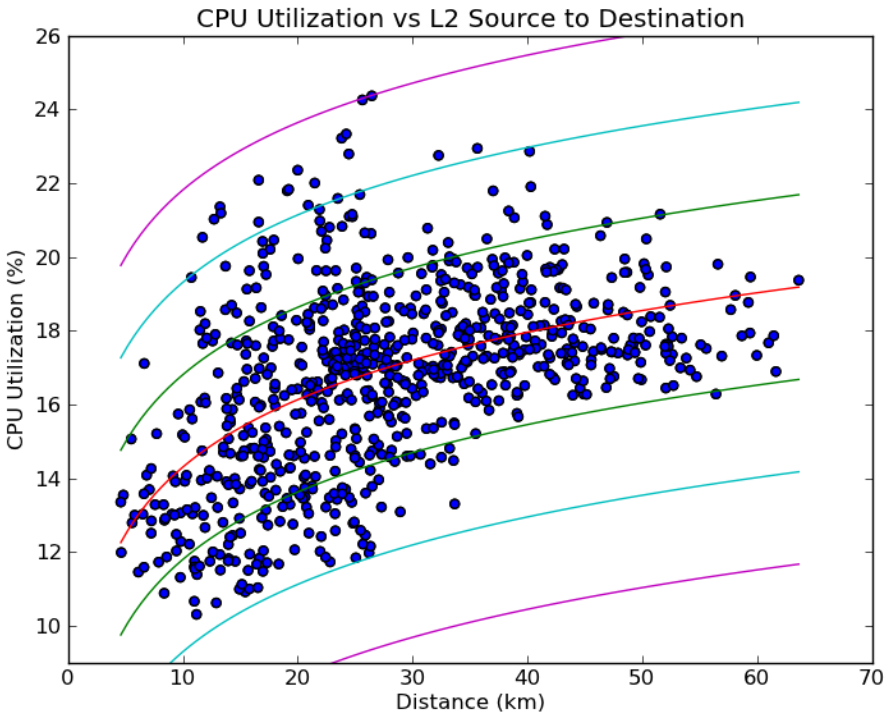

FIGURE 4.7: CPU USAGE VS REQUEST DISTACNE

$$y = 2.64 \cdot \mathrm{LOG}\, x + 10.01 \tag{4.1}$$

4.2.1.3 TIME

When plotting computation time it was noted that these times appeared to scale logarithmically when plotted against the L2 distance between source and destination nodes, this plot is included as Figure 4.8. In order to verify this observation a best fit line was produced via a least squares approximation, this equation is included as Equation 4.2 and marked as the red line on the plot. It was determined that although 89.59% of the data fell within one standard deviation of this fit, marked on the plot as green lines, there were a significant number of outliers. The top six outliers, circled on the plot, formed a cluster and the requests that generated these outliers were analyzed. It was found that these requests all had either a source or destination node in an area of the map with no charging station and the opposite node was on the other side of down-town Rochester. It is theorized that this condition led to an abnormally large number of paths for the A* algorithm to visit.

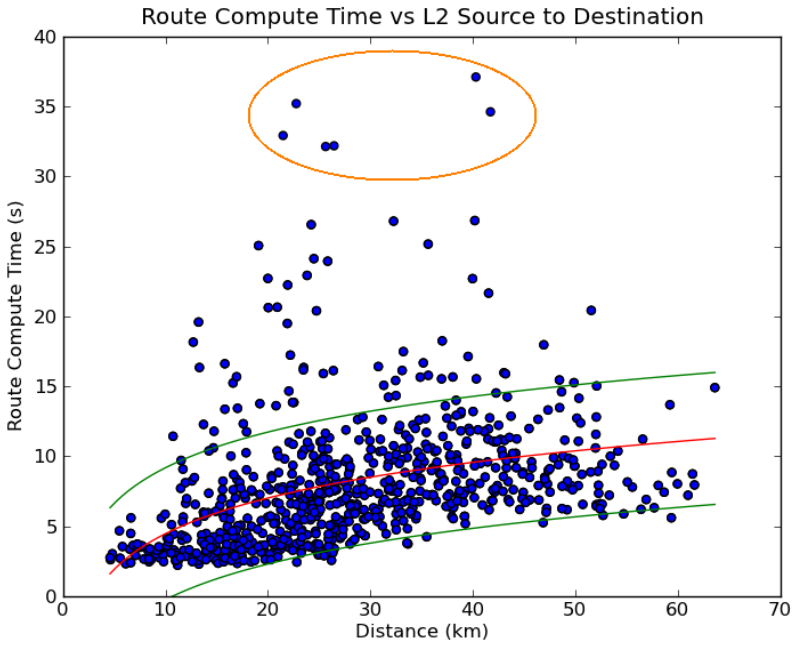

FIGURE 4.8: ROUTE COMUTE TIME VS. REQUEST DISTANCE

$$y = 2.64 \cdot \mathrm{LOG}\, x + 10.01 \tag{4.2}$$

From this data it is obvious that the computation time scales logarithmically, making the system highly scalable for use with a larger map size. The computation time of the outliers could be reduced through cloud elasticity, assigning more cores to the system for these requests. Due to the fact that both the RAM and CPU used to achieve these times are significantly greater than that found in a GPS routing device, it is easy to see why a cloud was selected to host the system rather than a personal GPS routing device.

At the end of the simulation the total amount of time spent in each subsystem was analyzed and is visualized in Figure 4.9. The traffic server computation time section, as well as the garbage sections, including such things as set-up, memory allocation and garbage collection, were both negligible. The prototype system spent roughly 11% of its time waiting for the station server due to the fact that the charge station database had to be queried and the results processed. Because of RMI overhead, roughly 7.5% of the total computation time was spent communicating between subsystems. If a custom application layer protocol was developed, thus eliminating the RMI overhead, the communication time would be reduced significantly. The rest of the computation time was spent routing and selecting charge stations; this is where any effort to optimize the system should be focused.

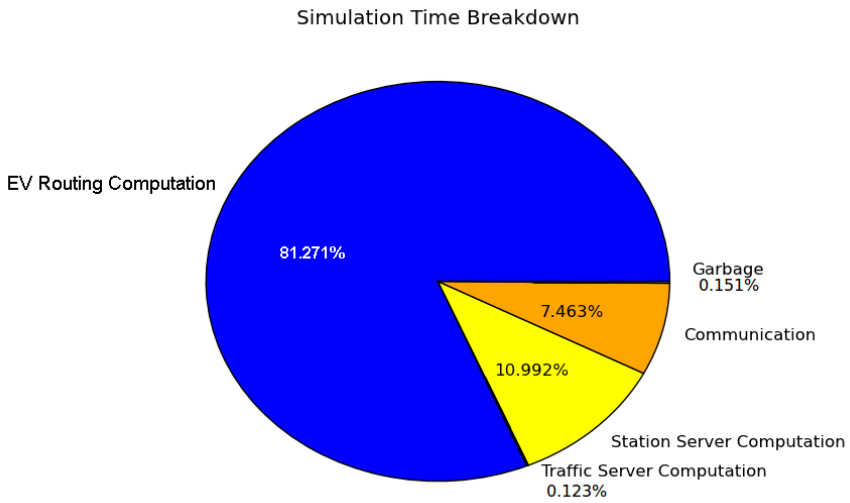

FIGURE 4.9: ROUTE COMPUTE TIME BREAKDOWN

4.2.1.4 USER LATENCY

In order to determine how the system handles request saturation, the average request waiting time was plotted as Figure 4.10. The average request waiting time is defined as the number of simulated seconds between when a request is received and the time the system begins to process the request. The simulation time was updated by adding the actual computation time to the current simulation time. If the next request was sent after the current time, there is no delay and the request's received time is the new simulation time. As shown in the plot, the system reaches saturation somewhere around 200 requests per hour, before this point the waiting time is negligible (no more than 30 seconds), after the waiting time increases at roughly a rate of 10 seconds per request. In order increase the saturation point and decrease the rate of waiting time post-saturation, the system could be multi-threaded in order to process more than one request simultaneously.

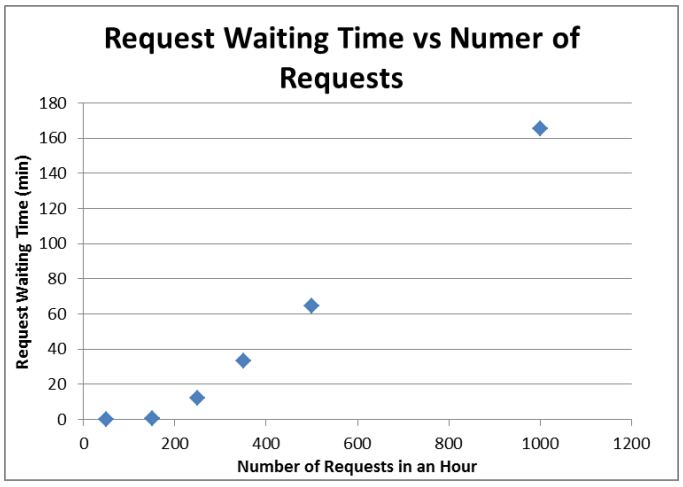

FIGURE 4.10: REQUEST WAITING TIME

4.2.2 VARIATION IN TIME OF DAY

Simulations were run in order to determine how traffic conditions affects a user's ability to charge their vehicle, and to determine the effectiveness of traffic data in route calculation. The Traffic Modeler program, discussed in section 3.2.3.1, provided a full day's worth of realistic traffic, and SUMO simulations were run producing appropriate traffic for three times of the day: rush hour (7am-8am), mid-morning (11am-12 noon), and late at night (9pm-10pm). Furthermore, the Energy Index (EI) of requests were modified in order to emulate energy expenditure under different conditions: during rush hour it was decreased by 10% in order to emulate stop and go traffic, and during late-nigh it was increased by 10% in order to emulate open roads. These changes were based on the difference in EV ranges presented by the same consumer reports discussed in section 3.2.4.1.

TABLE 4.5

TIME OF DAY SIMULATION RESULTS

Late Night			
Set	success	fail	not needed
A	142	0	8
B	136	8	6
C	134	9	7
D	138	6	6
E	145	2	3
Mid-Afternoon			
Set	success	fail	not needed
A	140	6	4
B	132	15	3
C	128	21	1
D	136	11	3
E	142	6	2
Rush Hour			
Set	success	fail	not needed
A	123	25	2
B	115	34	1
C	107	43	0
D	118	31	1
E	125	23	2

Five different sets of requests were generated and an equal number of traffic simulations for each time of the day, the resulting success, fail, and charge not needed counts included in Table 4.5 The success/fail ratio was analyzed and is included in the above figures. Figure 4.11 shows that, as predicted, nighttime is best time to charge an

EV, while Figure 4.12 shows that mid-afternoon is a close second. Both of these times had a relatively low failure rate indicating a good distribution of charging stations. However, as indicated by Figure 4.13, one fifth of the requests made during rush hour failed. This is obviously the worst case condition and as such was used for all future simulations.

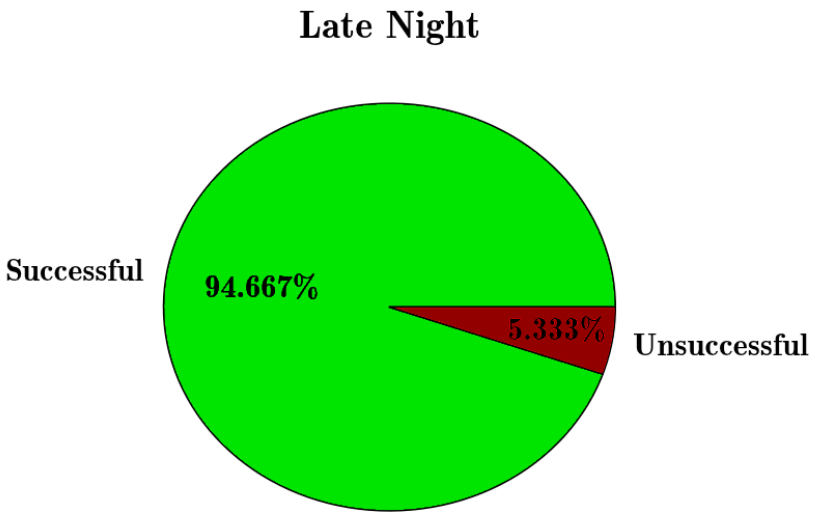

FIGURE 4.11: LATE NIGHT SIMULATION STATISTICS

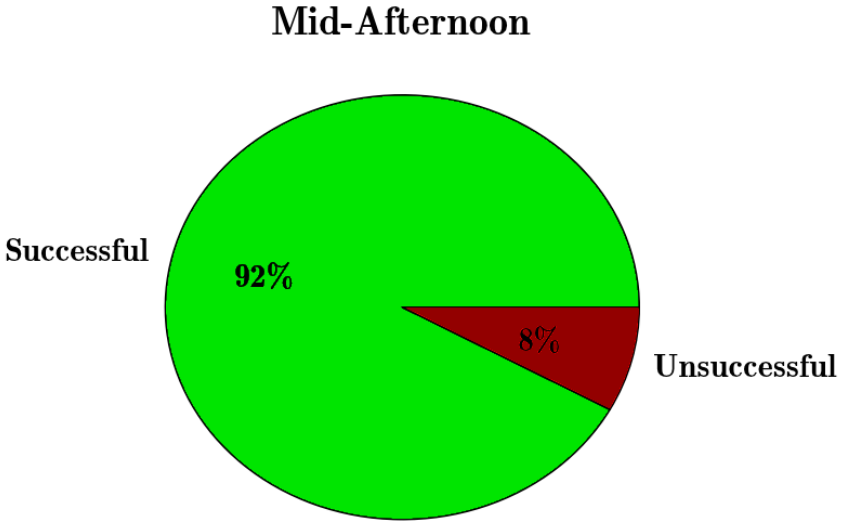

FIGURE 4.12: MID-AFTERNOON SIMULATION STATISTICS

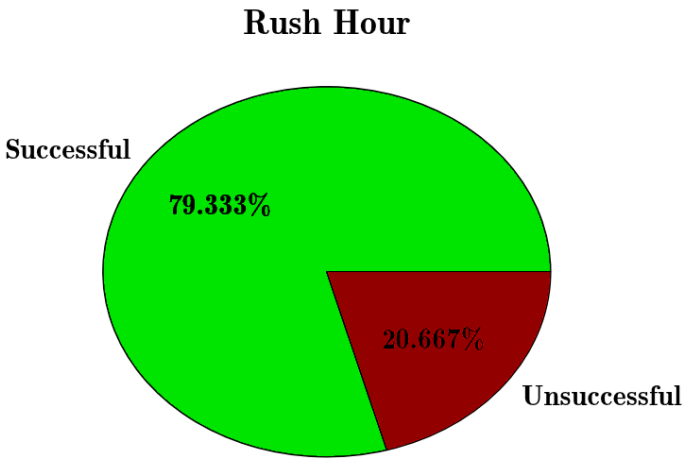

FIGURE 4.13: RUSH HOUR SIMULATION STATISTICS

Average additional travel time for routes calculated both using traffic data and without are plotted as Figure 4.14. In all cases the inclusion of traffic data produced a superior route. The difference in travel time is most noticeable during rush hour due to traffic congestion. Travel times are plotted as a fraction of the direct route between the source and the destination in order to identify routes which are significantly longer than a direct route. For example, an extra 20 minutes is more significant if the source and destination are 5 minutes apart than if they are an hour apart. The increase in the percentage using traffic data as compared to without is due to the fact charging stations are located in busy areas vehicles would not travel otherwise. I should be noted however that the algorithm can still generally avoid high density roads resulting in only a small percentage increase.

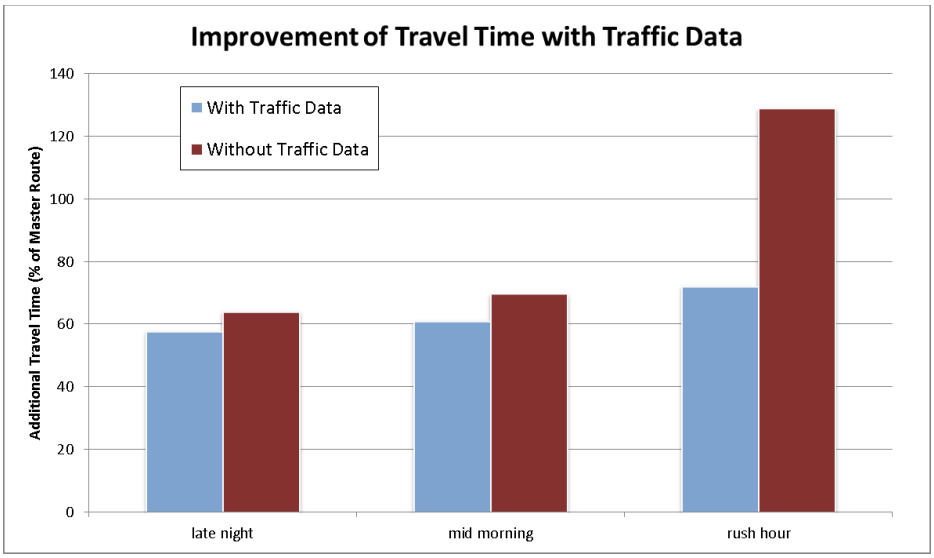

FIGURE 4.14: IMPROVEMENTS WITH TRAVEL TIME

4.2.3 ADDING PLAUSIBLE CHARGING STATIONS

The charging stations in the Rochester area were not evenly distributed, leading to a high fail rate during rush hour as well as a significant detour distance. What was shown so far was based on the 6 existing charging stations in the Rochester area. In this set of simulations, charging stations were introduced at plausible locations including city hall, museums, parks, etc., and the resulting success/failure rate and charge station distribution were observed. Worst case (Rush Hour) conditions were used in this test. All six stations are depicted in the map included as Figure 4.15; the stations marked with a charge and with a letter next to it are real level 2 charging stations in the Rochester area, stations marked in blue with a number are artificially introduced Level 2 stations. Stations were introduced two at a time in order to assess their impact, if a station had negligible impact it was relocated and re-assessed before the next pair was introduced. Figure 4.16 denotes the location of request origin and destinations for the dataset used in this experiment. The number of origins in the upper left hand corner, not within range of a charging station, should be noted.

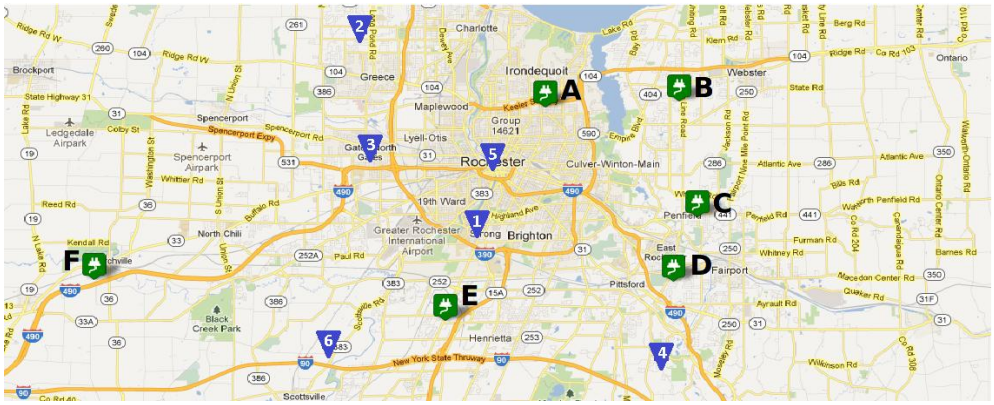

FIGURE 4.15: CHARGEING STATION LOCATIONS

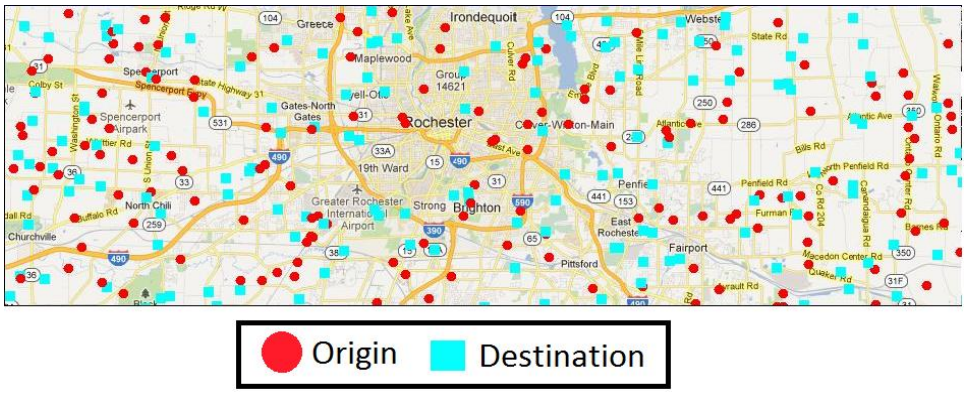

FIGURE 4.16: ORIGIN AND DESTINATION LOCATIONS

Figure 4.17 and Figure 4.19 depicted the base case scenario, that is to say only real charging stations were used in this simulation. As with pervious rush hour simulations there were a large number of routing failures, and charging stations E and F

were selected an unproportionate number of times; these stations will get overwhelmed with a high EV penetration rate. On the other hand, Figure 4.18 depicts a fairly even distribution in the distances vehicles must travel to reach each station.

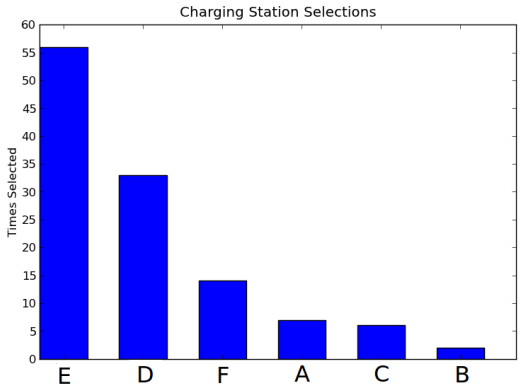

FIGURE 4.17: NO ADDITIONAL STATION SELECTIONS

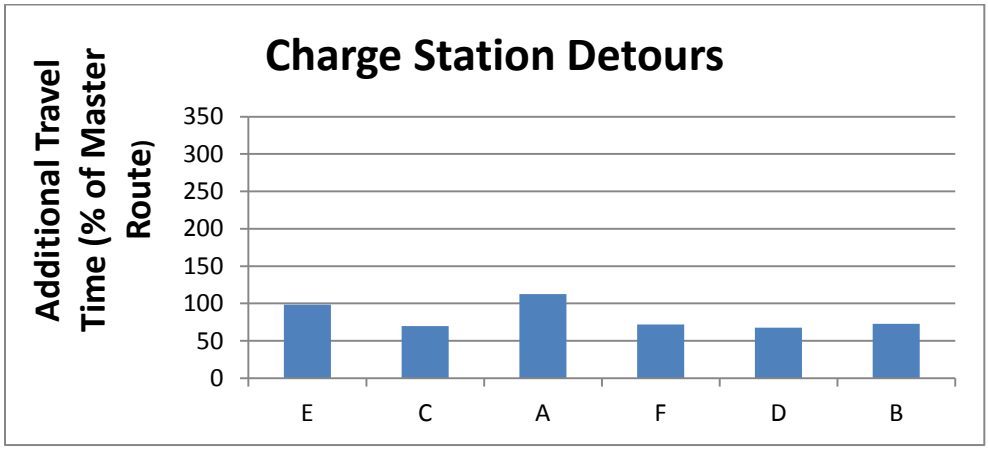

FIGURE 4.18: NO ADDITIONAL STATIONS DETOUR DISTRIBUTION

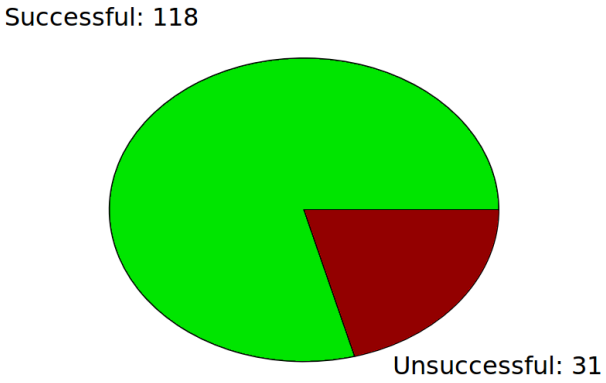

FIGURE 4.19: NO ADDITIONAL STATIONS SUCCESS RATE

Stations were added in areas without other charging stations in the immediate vicinity. A station was added near the center of the map at the University of Rochester administration building (station 1) and another in the corner at the Greece Town Hall (station 2). It should be noted that the larger total number of selections is due to the higher success rate depicted in Figure 4.22, and that this success rate does not change when adding any further stations. As shown in Figure 4.20, Station 1 received a large portion of the request previously held by Station E, while station 2 seems to be mostly selected by the newly successful requests. The most popular stations will still get overwhelmed, although not to the extent in the previous simulation. Figure 4.21 shows that vehicles have to go a significant distance out of their way to reach stations one and two; this is because these stations are on the edge of the ranges of the newly successful requests and there are no other stations nearby. These newly successful requests are the ones with origins in the upper left corner of Figure 4.16.

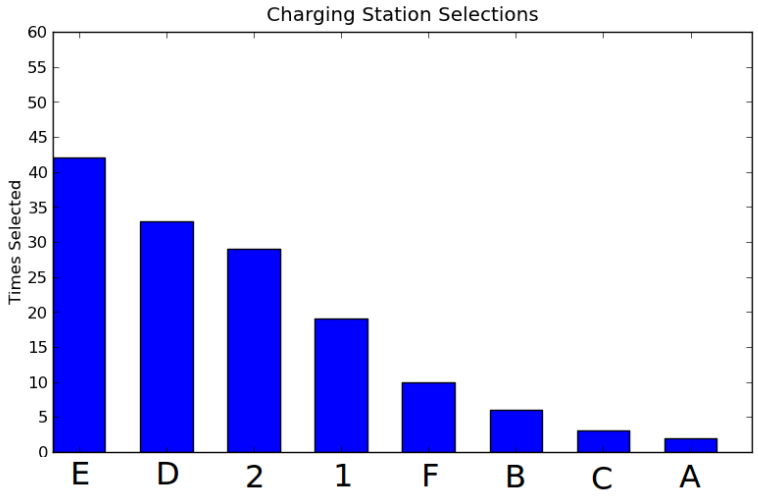

FIGURE 4.20: ADDITIONAL STATIONS SELECTION (1-2)

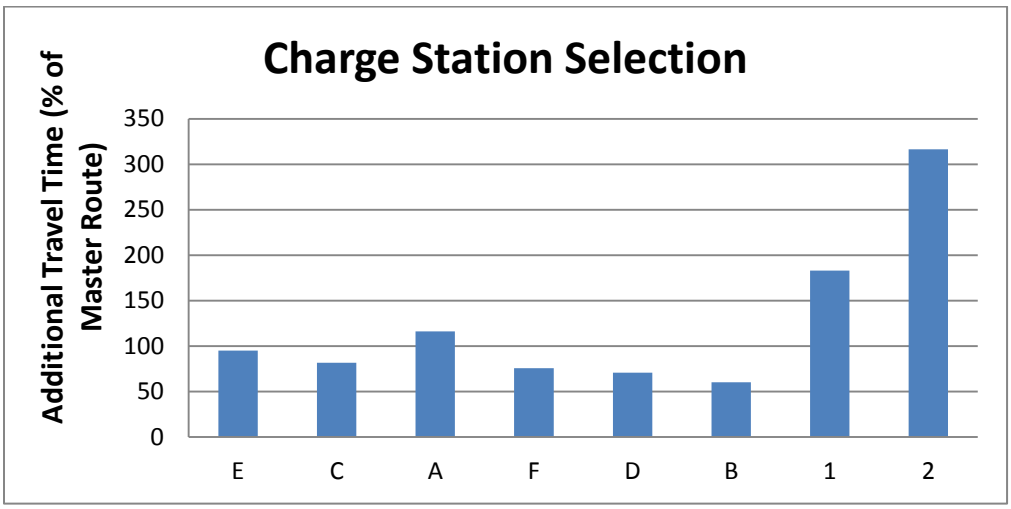

FIGURE 4.21: ADDITIONAL STATIONS DETOUR DISTRIBUTION (1-2)

FIGURE 4.22: ADDITIONAL STATIONS SUCCESS RATE

Stations were added at the Gates Fuel Depot (Station 3) and Powder Mills Park (station 4) and then Rochester City Hall (station 5) and the Chili Country Club (station 6). Although these stations did not increase the success/fail rate, they more evenly distributed than the station selection as depicted in Figure 4.23, Figure 4.24, Figure 4.25, and Figure 4.26.

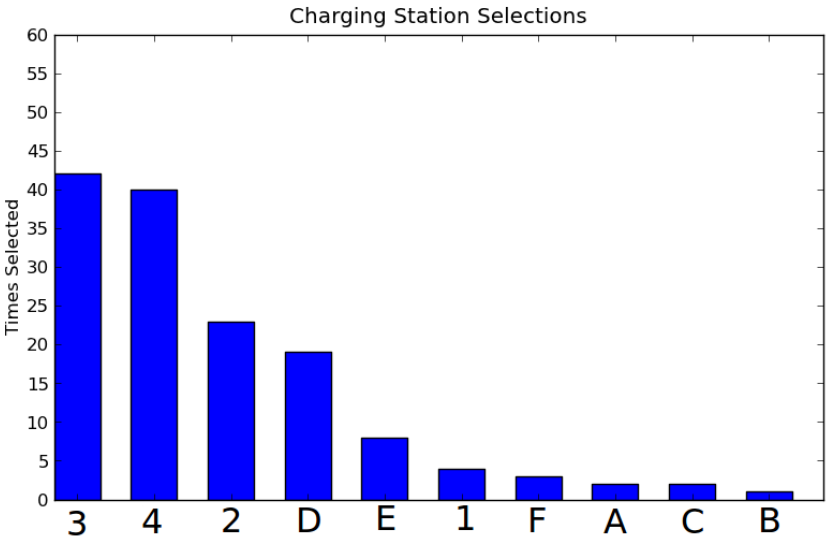

FIGURE 4.23: ADDITIONAL STATIONS SELECTION (1-4)

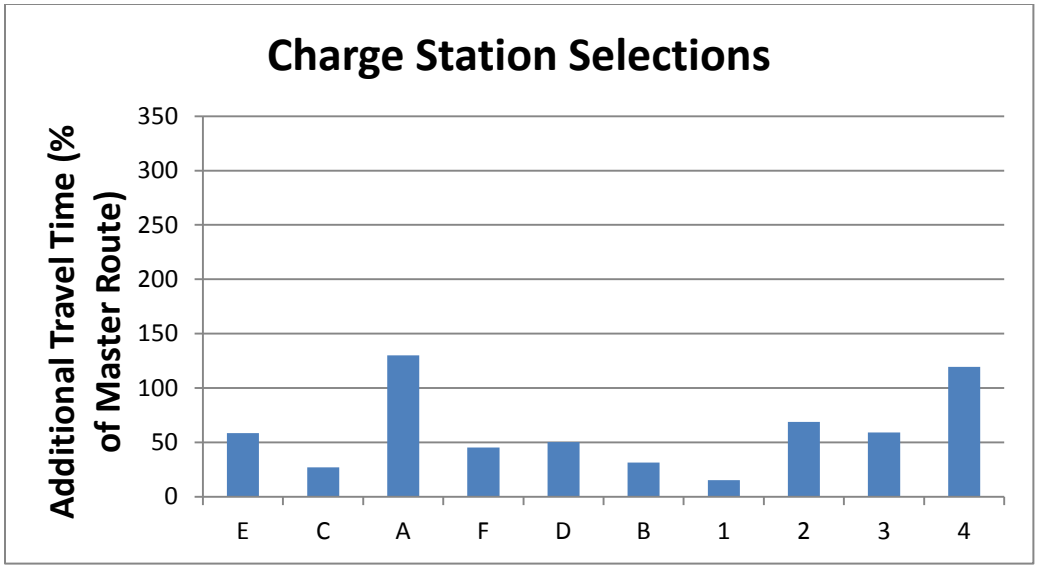

FIGURE 4.24: ADDITIONAL STATIONS DETOUR DISTRIBUTION (1-4)

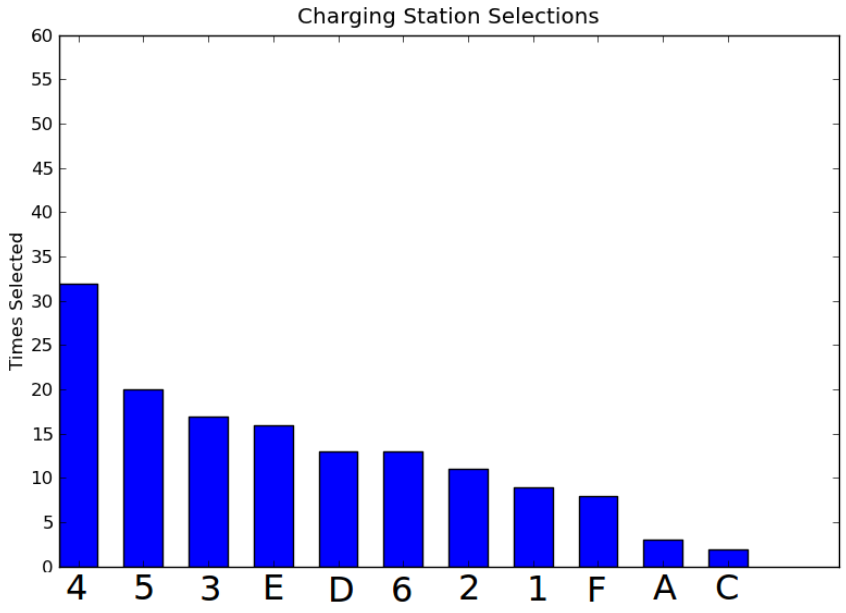

FIGURE 4.25: ADDITIONAL STATIONS SELECTIONS (1-6)

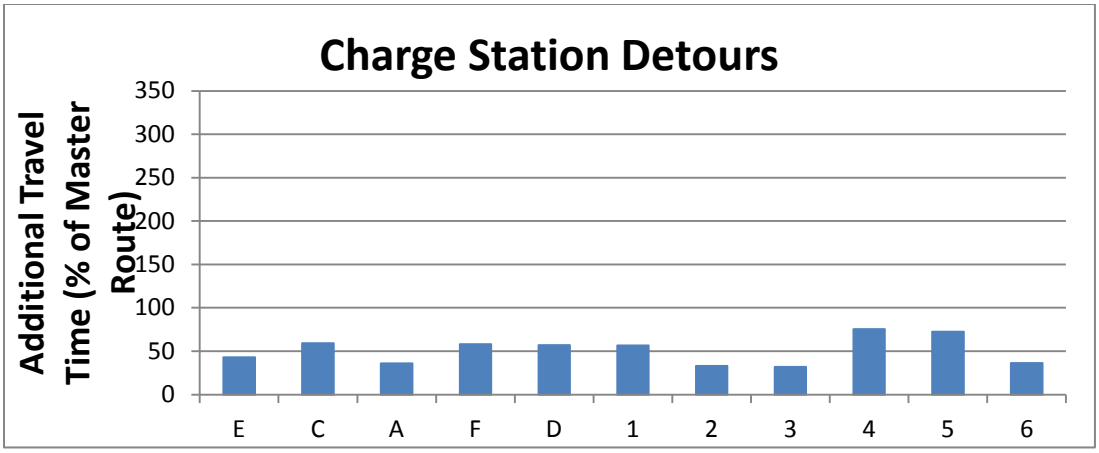

FIGURE 4.26: ADDITIONAL STATIONS DETOUR DISTRIBUTION (1-6)

The average additional travel time required to reach each station is plotted in Figure 4.27. In general, as more stations were added the detour distance decrease; the increase when adding stations 1 and 2 is due to the increase in the success rate and the fact that most of the additional selections were on the edges of the EVs range. As more stations were added, their benefit in terms of detour distance decreased, due to over-population.

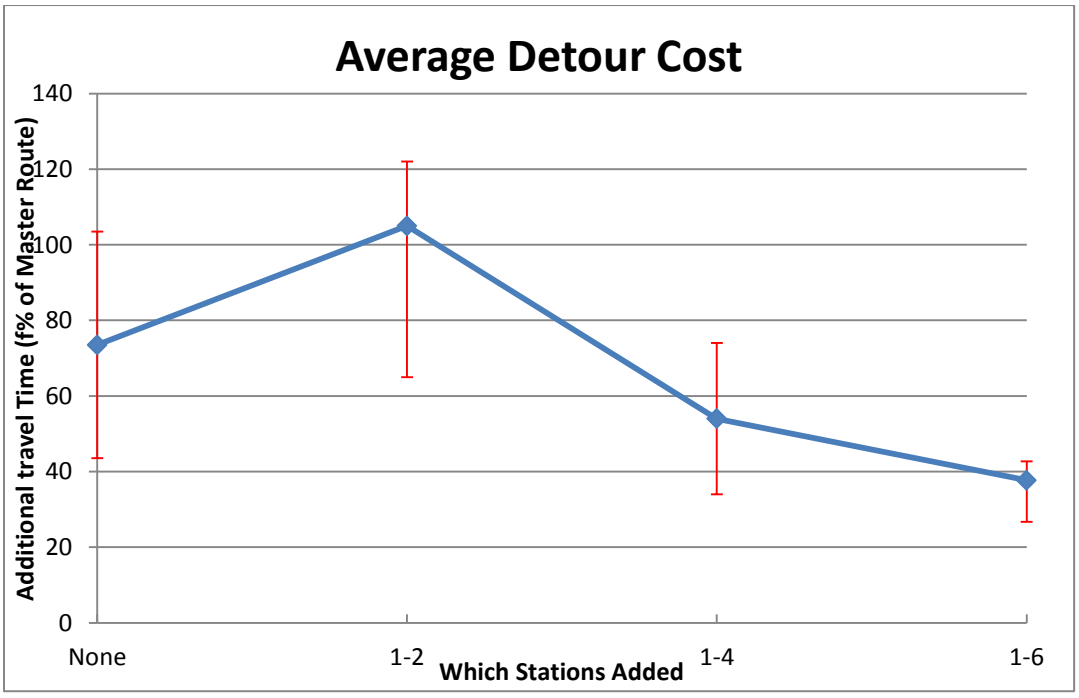

FIGURE 4.27: AVERAGE DETOUR COST WITH ADDITIONAL STATIONS

These results show that as more charging stations are added, they will allow more EVs to successfully charge, will distribute the EV charging more evenly, and will decrease the EV's additional travel time. The benefit of adding more charge stations decreases rapidly for such a small EV penetration level, however with a significant

increase in penetration level these benefits should remain constant with the addition of many additional stations.

In a real world scenario there would not be just 6 charging stations constructed. A simulations was run on a more extreme case where 40 charging stations where randomly placed in the locations depicted in Figure 4.28. As depicted in Figure 4.29 and Figure 4.30, the trends described earlier continued, including a reduction is peak station selection and total travel time as well as a increase in distribution informality. This verified this system could be used in larger scenarios in the future.

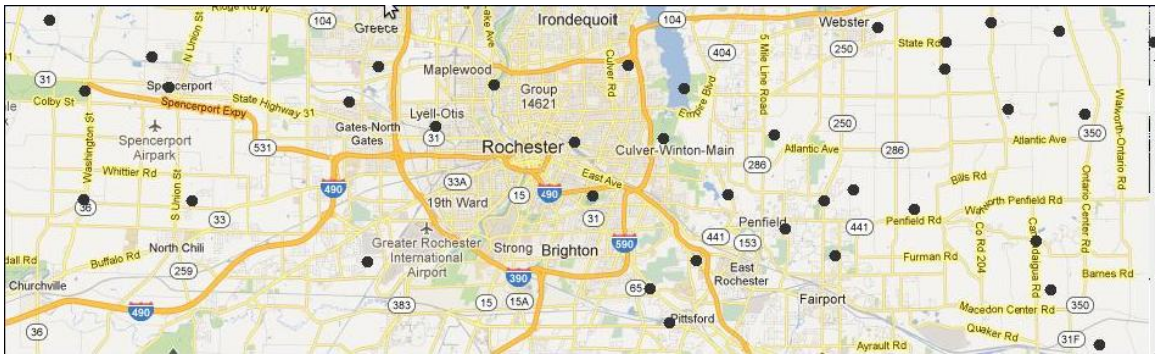

FIGURE 4.28: RANDOM STATION LOCATIONS

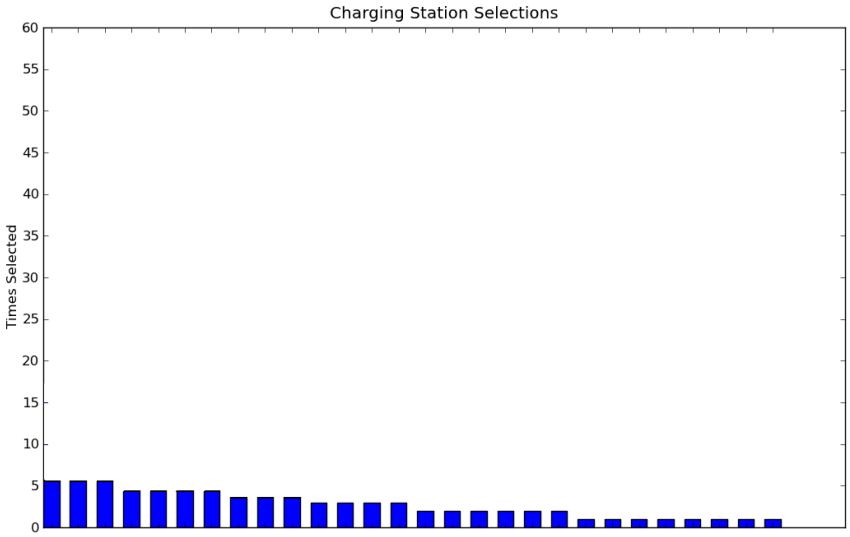

FIGURE 4.29: EXTREME CASE STATION SELECTION

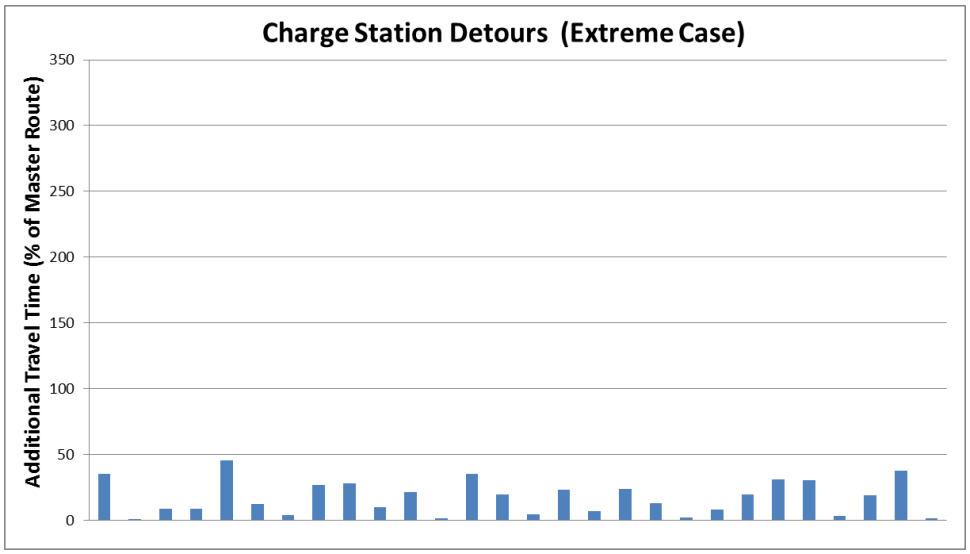

FIGURE 4.30: EXTREME CASE DETOUR DISTRIBUTION

4.2.4 ROUTING PREFERENCE VALIDATION

In order to determine the extent to which selection preference plays in the selection of a charge station, the average detour cost and electricity costs were plotted while varying theses values. In the current algorithm, the alpha factor places more emphasis on additional travel time, while the beta factor places more emphasis on the cost of electricity at the charging stations. Rush hour conditions and the additional charge stations introduced in the previous section were used as test data. As was expected and shown by Figure 4.31 and Figure 4.32, the detour cost increased and electricity cost decreased as more preference was given to electricity cost. These changes did not occur linearly however, around .6/.4 there was a significant change in slope.

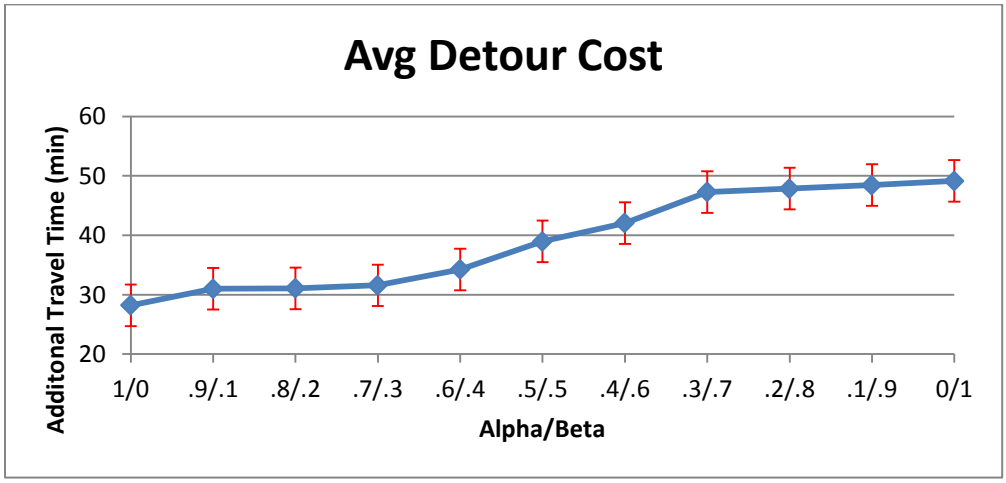

FIGURE 4.31: ALPHA/BETA DETOUR COST

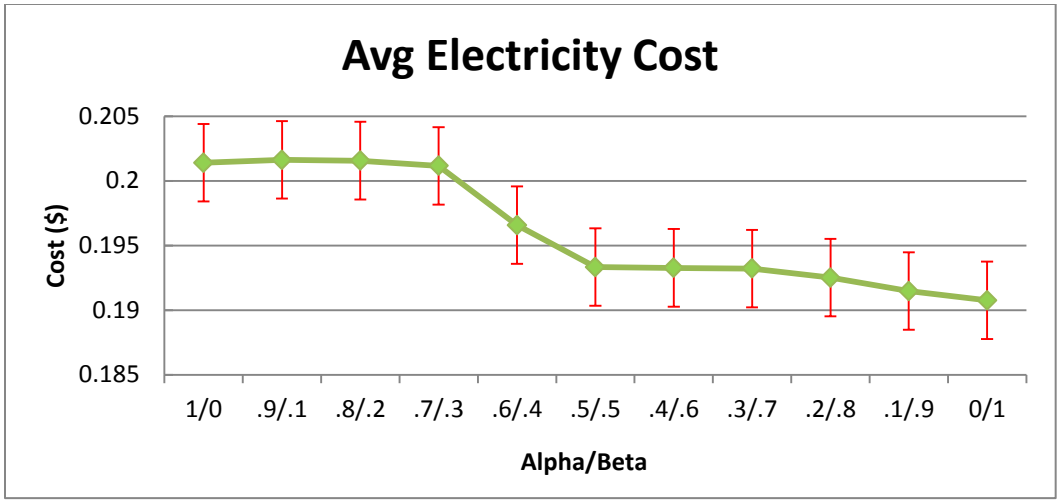

FIGURE 4.32: ALPHA/BETA ELECTRICITY PRICE

Because the electricity cost was generated and thus known to be a standard Gaussian Distribution, the only cause of this inconsistency could be the distribution of the additional travel times. These distances were used to generate the histogram included as Figure 4.33. The large standard deviation of this distribution is believed to have caused the change in slope while the shifted mean of the distribution is believed to have shifted the inconsistency away from .5/.5. The inconsistency is believed to be more prominent in Figure 4.32, due to the smaller deviation in electricity costs.

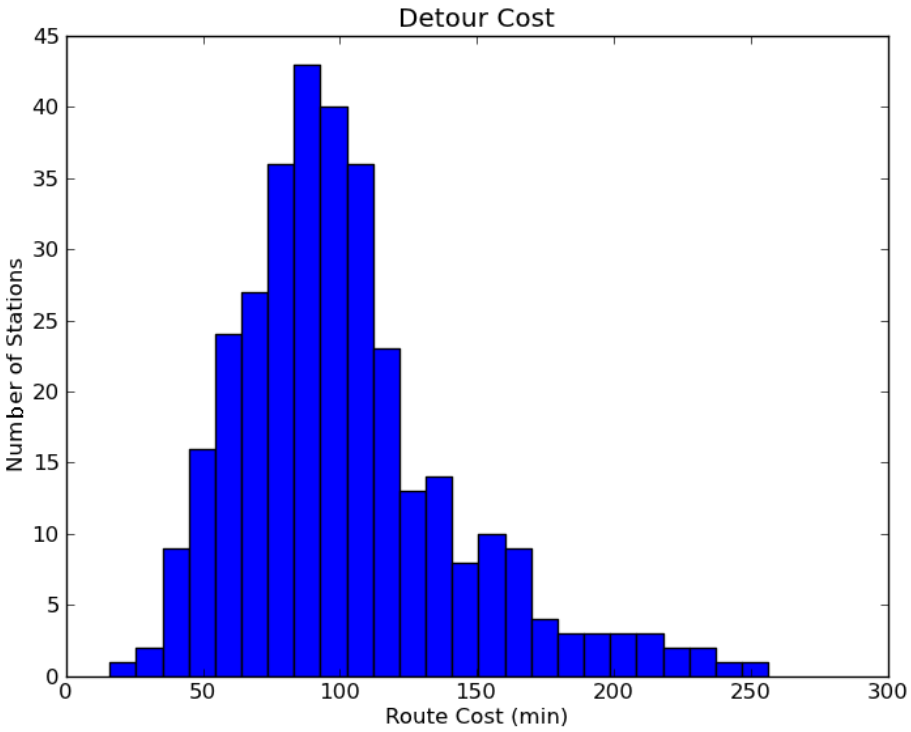

FIGURE 4.33: DETOUR COST DISTRIBUTION

In an attempt to find a "sweet spot" for the metric and simultaneously optimize both distance and cost, both of the previous graphs were normalized between zero and one and then plotted on the same graph. As depicted in Figure 4.34 simultaneous optimization occurred around .55/.45; however, both values were only 40% of their maximum. Initial slope decrease occurs around a preference of .7, this value does not produce a significant reduction of the preferred metric while maximizing the other, making it the ideal selection preference on both ends of the spectrum.

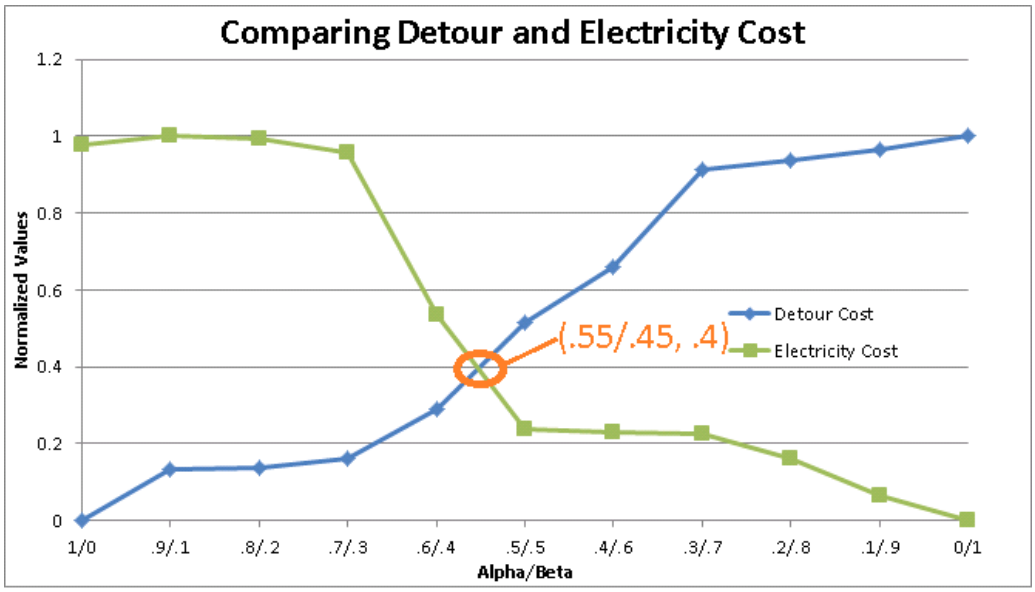

FIGURE 4.34: SIMULTANEOUS OPTIMIZATION

5 FUTURE WORK/CONCLUSION

Today's power grid is over 70 years old and aging poorly. This combined with the introduction of Plug in Hybrid Electric Vehicles (PHEVs) and pure Electric Vehicles (EVs) could potentially mean the failure of the US power grid, due to the fact that charging each of these vehicles overnight demands roughly the same amount of power as a family home. Eventually, consumers will not be content with charging their vehicles overnight and will demand the convenience and availability currently found with petroleum-fueled vehicles. Current models of "quick charge stations" require the same amount of power per car for a 10-minute charge as is consumed by 140 homes. That amount of power is simply not available on demand with the current power grid model. Remodeling or replacing the current power grid would be expensive, time consuming, and may not be effective. The work presented in this thesis includes a prototype system that routes EVs to the charging station which is both the least harmful to the electric grid and has the least amount of additional travel time, taking advantage of data that is predicted to be available within a single cloud data space in the near future.

Previous works have not attempted to use data gathered from existing systems, have not incorporated current traffic conditions into routing, and have not developed a prototype system to serve as a proof-of-concept; all of which was done in this thesis. If a more thorough case study is required and there is time to spare the work presented [34] in should be used. To plan longer trips where time is not a significant issue, the work presented in [36] should be used. Lastly, to plot an optimal trip using real time traffic information, the work presented in this thesis should be used. In an ideal system the work presented in [36] and the work presented in this thesis would be merged into a single system.

The prototype system presented serves as a proof-of-concept, in order to demonstrate the viability of the cloud as a computation platform for such a system, the benefit from a shared data space consisting of data gathered from sensor networks/the smart grid, and how such a system could be used in planning the location of future charging stations. The system uses the open source program Traveling Salesman as a base in order to implement a router which uses simulated real-time traffic conditions and current conditions at each charging station in order to select an optimal charging station for each EV to issue a request. The sub systems responsible for reporting traffic conditions and station information are housed in separate Virtual Machines within a small cloud system in order to emulate how this prototype would actually be deployed.

Included in this document are the results of a series of stress tests that show that the system is highly scalable but cannot be implemented on an average GPS routing device. The system was not designed for optimal usage of CPU power or RAM; future work could include optimizations designed at reducing system requirements. The system currently processes the requests linearly as they are received, resulting in a significant processing delay over a rate of 250 requests per hour. This delay could be significantly reduced if a different processing scheme was implemented, such as a multi-threaded job

queue scheme. Also included in this document are the results of a case analysis of possible locations for additional charging stations in Rochester, NY. If a more detailed EV model, a model for charging the vehicle, and a model representing the electrical grid were to be used, a more detailed and accurate report could be generated.

Additional future work includes incorporating additional factors into the routing metric, tying the router into the traffic simulator so that EV charging would affect traffic flow, a more accurate model of the estimated distance the EV can travel, and a custom communication protocol in order to eliminate the delay introduced by Java's RMI protocol.

6 BIBLIOGRAPHY

[1] A. Faza, S. Sedigh, and B. McMillin, "Reliability Modeling for the Advanced Electric Power Grid," *Computer Safety, Reliability, and Security*, vol. 1, no. 573, pp. 370–383, 2007.

[2] R. A. Waraich, "Plug-in Hybrid Electric Vehicles and Smart Grid: Investigations Based on a Micro-Simulation," *The 12th International Conference of the International Association for Travel Behavior Research*, pp. 1–23, 2009.

[3] M. Geske, P. Komarnicki, and M. Stotzer, "Modeling and Simulation of Electric Car Penetration in the Distribution Power System – Case Study," *Electric Power Magazine 2010*, pp. 1–6, 2010.

[4] M. M. Hassan, B. Song, and E. N. Huh, "A framework of sensor-cloud integration opportunities and challenges," *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, pp. 618–626. 2009

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[6] B. Rochwerger and D. Breitgand, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no 5, pp. 1–17, 2009.

[7] D. Krajzewicz, G. Hertkorn, P. Wagner, and C. Rössel, "SUMO (Simulation of Urban MObility) An open-source traffic simulation Car-Driver Model," *Proceedings of the 4th Middle East Symposium on Simulation and Modeling* , pp. 183–187, September 2002

[8] K. Sharpe and C. Cook, "OpenChargeMap." [Online]. Available: http://openchargemap.org/about/. [Accessed: 10-Sep-2012].

[9] T. Winkler, P. Komarnicki, G. Mueller, G. Heideck, M. Heuer, and Z. A. Styczynski, "Electric vehicle charging stations in Magdeburg," *Proceedings of IEEE Vehicle Power and Propulsion Conference 2009*, pp. 60–65., 2009

[10] T. Ricker, "North America's first public-use quick-charge station opens in Portland," *engadget*, Aug-2010. [Online]. Available: http://www.engadget.com/2010/08/06/north-americas-first-public-use-quick-charge-station-opens-in-p/.

[11] S. Letendre and R. A. Watts, "Effects of plug-in hybrid electric vehicles on the Vermont electric transmission system," *Proceedings of Transportation Research Board Annual Meeting, Washington DC, 2009*, no. 802, pp. 11–15, 2009

[12] K. Yunus and D. L. Parra, “Distribution grid impact of Plug-In Electric Vehicles charging at fast charging stations using stochastic charging model,” *IEEE Conference on Power Electronics and Applications 2011*, pp. 1-11 2011.

[13] L. Dickerman and J. Harrison, “A new car, a new grid,” *Power and Energy Magazine,* April 2010, pp. 55–61, 2010.

[14] C. C. Chan, “The State of the Art of Electric, Hybrid, and Fuel Cell Vehicles,” *Proceedings of the IEEE*, vol. 95, no. 4, pp. 704–718, Apr. 2007.

[15] “OnStar Looking to Make the Smart Grid Smarter.” [Online]. Available: http://media.gm.com/content/media/us/en/onstar/news.detail.html/content/Pages/news/us/en/2012/Feb/0202_onstar.html#.UFCyQOc3EgY.mendeley. [Accessed: 12-Sep-2012].

[16] R. Maia, M. Silva, and R. Araujo, “Electric vehicle simulator for energy consumption studies in electric mobility systems,” *Proceedings of Integrated and Sustainable Transportation Systems 2011,* pp. 227 - 232, 2011

[17] S. Kaplan, “Electric power transmission: background and policy issues,” *US Congressional Research Service, April*, 2009.

[18] H. Farhangi, “The path of the smart grid,” *Power and Energy Magazine, February 2010*, pp. 18-28, 2010

[19] “A vision for the modern Grid”, National Energy Technology Laboratory , United States department of energy, March 2007 [online] Available: http://www.netl.doe.gov/moderngrid/docs/A Vision for the Modern Grid_Final_v1_0.pdf. Retrieved 2011-6-27.

[20] W.-D. Xie and W. Luan, “Modeling and simulation of public EV charging station with power storage system,” *Proceedings of the International Conference on Electric Information and Control Engineering 2011*, pp. 2346–2350, Apr. 2011.

[21] A. Schroeder and T. Traber, “The economics of fast charging infrastructure for electric vehicles,” *Energy Policy*, vol. 43, pp. 136–144, Jan. 2012.

[22] K. Nansai, S. Tohno, M. Kono, M. Kasahara, and Y. Moriguchi, “Life-cycle analysis of charging infrastructure for electric vehicles,” *Applied Energy*, vol. 70, no. 3, pp. 251–265, Nov. 2001.

[23] L. E. Y. Mimbela and L. A. Klein "A Summary of Vehicle Detection and Surveillance Technologies Used in Intelligent Transportation Systems," *The Vehicle Detector Clearinghouse 2000* [online] Available: http://www.fhwa.dot.gov/ohim/tvtw/vdstits.pdf Retrieved 2011-7-02

[24] L. Figueiredo, I. Jesus, J. a. T. Machado, J. R. Ferreira, and J. L. Martins de Carvalho, "Towards the development of intelligent transportation systems," *Proceedings of the IEEE Intelligent Transportation Systems Conference 2001*, no. 81, pp. 1206–1211, 2001.

[25] F. J. Martinez, C. K. Toh, J. C. Cano, C. T. Calafate, and P. Manzoni, "A survey and comparative study of simulators for vehicular ad hoc networks (VANETs)," *Wireless Communications and Mobile Computing*, vol. 11, no. 7, pp. 813–828, 2011.

[26] M. Chuah and F. Fu, "Performance study of robust data transfer protocol for VANETS," in *Mobile Ad-hoc and Sensor Networks*, pp. 377-391, 2006.

[27] J. Jeong, S. Guo, Y. Gu, T. He, and D. Du, "TBD: Trajectory-Based Data Forwarding for Light-Traffic Vehicular Networks," *Proceedings of the IEEE International Conference on Distributed Computing Systems 2009*, pp. 231–238, Jun. 2009.

[28] M. Abuelela and S. Olariu, "Taking VANET to the clouds," *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia 2010*, pp. 6–13, 2010.

[29] P. E. Hart and J. Nils, "Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, no. 2, pp. 100–107, 1968.

[30] J. Ambrose and D. Bukovsky, "Developing a travel route planner accounting for traffic variability," *Proceedings of the Systems and Information Engineering Design Symposium*, pp. 264–268, 2009.

[31] B. Fleischmann, M. Gietz, and S. Gnutzmann, "Time-Varying Travel Times in Vehicle Routing," *Transportation Science*, vol. 38, no. 2, pp. 160–173, May 2004.

[32] W. Services, "Satellite Weather Information and Intelligent Transportation System," *Environment*, vol. 130, no. 10, pp. 1–16, 2004.

[33] D. Wilkie, J. van den Berg, M. Lin, and D. Manocha, "Self-Aware Traffic Route Planning," *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pp. 1521–1527, 2011.

[34] S. Bessler and J. Grønbæk, "Routing EV Users Towards an Optimal Charging Plan," *Proceedings of the Electrical Vehicle Symposium*, pp. 1–8, 2012.

[35] O. Worley, D. Klabjan, and T. M. Sweda, “Simultaneous vehicle routing and charging station siting for commercial Electric Vehicles,” Proceedings of the *IEEE International Electric Vehicle Conference 2012*, pp. 1–3, Mar. 2012.

[36] Y. Kobayashi and N. Kiyama, “A route search method for electric vehicles in consideration of range and locations of charging stations,” *Proceedings of the Intelligent Vehicles Symposium 2011*, pp. 920–925, 2011.

[37] S. Dornbush and a. Joshi, “StreetSmart Traffic: Discovering and Disseminating Automobile Congestion Using VANET’s,” *Proceedings of the 65th Vehicular Technology Conference* , pp. 11–15, Apr. 2007.

[38] W. Grosso and P. O. Reilly, *Java RMI*, October. O’Reilly Media, Inc., 2001, p. 545.

[39] M. Wolschon, “Traveling Salesman - the application.” [Online]. Available: http://sourceforge.net/apps/mediawiki/travelingsales/index.php?title=Main_Page. [Accessed: 02-Feb-2012].

[40] STMicroelectronics, “STA2062 Specifications.” [Online]. Available: http://www.st.com/internet/automotive/product/194027.jsp.